(12) DEMANDE INTERNATIONALE PUBLIÉE EN VERTU DU TRAITÉ DE COOPÉRATION EN MATIÈRE DE BREVETS (PCT)

(19) Organisation Mondiale de la Propriété Intellectuelle
Bureau international

(43) Date de la publication internationale
16 août 2001 (16.08.2001)

PCT

(10) Numéro de publication internationale
WO 01/59563 A1

(51) Classification internationale des brevets[7] :
G06F 9/445, H04L 29/06

(21) Numéro de la demande internationale :
PCT/FR01/00393

(22) Date de dépôt international : 9 février 2001 (09.02.2001)

(25) Langue de dépôt : français

(26) Langue de publication : français

(30) Données relatives à la priorité :
00/01661 10 février 2000 (10.02.2000) FR

(71) Déposant *(pour tous les États désignés sauf US)* : BULL CP8 [FR/FR]; 68, route de Versailles, B.P. 45, F-78430 Louveciennes (FR).

(72) Inventeurs; et

(75) Inventeurs/Déposants *(pour US seulement)* : URIEN, Pascal [FR/FR]; 4, rue du Ruisseau St Prix, F-78450 Villepreux (FR). BOUDOU, Alain [FR/FR]; 12, rue du Moulin, F-78930 Vert (FR). SIEGELIN, Christoph [DE/FR]; 32, rue Ginoux, F-75015 Paris (FR).

(74) Mandataire : BULL S.A.; Corlu, Bernard, PC58D20, 68, route de Versailles, F-78434 Louveciennes Cedex (FR).

(81) États désignés *(national)* : AU, CA, CN, JP, KR, SG, US.

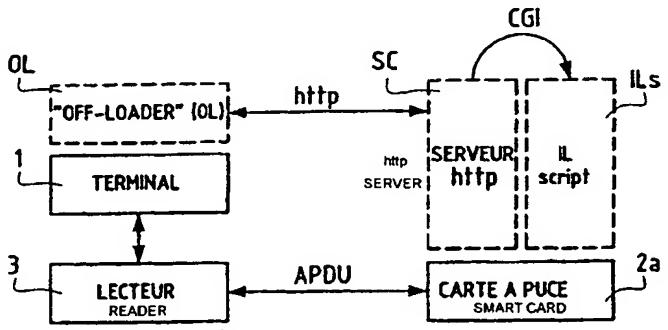
(84) États désignés *(régional)* : brevet européen (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR).

Publiée :
— *avec rapport de recherche internationale*
— *avant l'expiration du délai prévu pour la modification des revendications, sera republiée si des modifications sont reçues*

[Suite sur la page suivante]

(54) Title: METHOD FOR LOADING A SOFTWARE COMPONENT IN A SMART CARD, IN PARTICULAR APPLET

(54) Titre : PROCEDE DE CHARGEMENT D'UNE PIECE DE LOGICIEL DANS UNE CARTE A PUCE, NOTAMMENT DU TYPE DIT "APPLET"



(57) Abstract: The invention concerns a method for loading an applet in a smart card (2a), using two loading programmes, a so-called In-loader (IL), stored in the card, and Off-loader, respectively. The invention is characterised in that two specific communication protocol layers are provided, one in a terminal (1) hosting the card reader, the other in the card. Said layers include in particular intelligent agents enabling the card to provide a WEB client/server and a CGI gateway functional capability. The method comprises at least one step during which a HTTP request is sent to the card, to address a HTML page, a step which consists in retrieving parameterising data transported by a HTML form and a step which consists in executing a second loading programme (IL) using the CGI functional capability to load the applet.

(57) Abrégé : L'invention concerne le chargement d'une "applet" dans une carte à puce (2a), à l'aide de deux programmes de chargement, dits "In-loader" (IL), stocké dans la carte, et "Off-loader" (OL), respectivement. Selon l'invention, on prévoit deux couches protocolaires de communication spécifiques, l'une dans un terminal (1) hébergeant le lecteur de la carte, l'autre dans la carte. Ces couches comprennent notamment des agents intelligents permettant à la carte d'offrir une fonctionnalité de client/serveur "WEB" et de passerelle "CGI". Le procédé comprend au moins une étape pendant laquelle une requête "HTTP" est envoyée à la carte, pour adresser une page "HTML", une étape de récupération de données de paramétrage véhiculées par un formulaire "HTML" et une étape d'exécution du second programme de chargement (IL), par mise en oeuvre de la fonctionnalité "CGI", pour charge l'"Applet".

WO 01/59563 A1

En ce qui concerne les codes à deux lettres et autres abréviations, se référer aux "Notes explicatives relatives aux codes et abréviations" figurant au début de chaque numéro ordinaire de la Gazette du PCT.

PROCEDE DE CHARGEMENT D'UNE PIECE DE LOGICIEL DANS UNE CARTE A PUCE, NOTAMMENT DU TYPE DIT "APPLET"

L'invention concerne un procédé de chargement d'une pièce de logiciel dans une carte à puce.

Elle s'applique plus particulièrement au chargement d'une pièce de logiciel appelée en français "appliquette", plus connue sous le terme anglo-saxon d' "applet". Il s'agit d'une application écrite en langage "JAVA" (marque déposée). Ces applications, généralement peu volumineuses, sont indépendantes des architectures des systèmes sur lesquelles elles sont implantées. Elles peuvent donc fonctionner sur n'importe quel système informatique, dans la mesure où celui-ci implémente le concept dit de "machine virtuelle JAVA" ("Java Virtual Machine"). Une application écrite en langage JAVA est en général traduite en un langage intermédiaire dit "Byte Code". La machine virtuelle JAVA précitée constitue un interprétateur de ce "Byte Code", de manière à être exécutée directement sur un système cible, hôte de ladite machine virtuelle.

Généralement, les architectures de système sur lesquelles tourne ce type d'applications sont du type dit client-serveur. Dans ce cas, on parle également de "servlet" pour une application stockée sur un système serveur et "applet" pour une application stockée sur un système client. Dans ce qui suit, on utilisera le terme "applet" de façon générique.

Des pièces de logiciel, se présentant sous la forme d' "applet" qui vient d'être rappelée, dans la mesure où la quantité de code n'est pas trop volumineuse, peuvent être stockées dans une mémoire non volatile présente sur une carte à puce, au même titre que toute autre application.

Aussi, le procédé selon l'invention est plus particulièrement concerné par un terminal ou station d'utilisateur muni d'un lecteur de carte à "puce".

Dans le cadre de l'invention, le terme "terminal" doit être compris dans un sens général. Le terminal précité peut être notamment constitué par un ordinateur personnel fonctionnant sous divers systèmes d'exploitation, tels WINDOWS ou UNIX (tous deux étant des marques déposées). Il peut être aussi constitué par une station de travail, un ordinateur portable ou un terminal de carte dit dédié.

Dans l'état actuel de la technique, le chargement d'une "applet" sur une carte à puce (encore appelé téléchargement) se fait grâce à deux programmes spécifiques. Ces programmes sont généralement connus sous les termes anglo-saxons "Off-Loader", pour le premier, et "In-Loader", pour le second. Le programme "Off-Loader" s'exécute sur le terminal et le programme "In-Loader" s'exécute dans la carte à puce. Les programmes de chargement "Off-Loader" et "In-Loader" communiquent entre eux au travers d'une liaison normalisée de type ISO 7816-3, protocole universellement retenu pour les communications entre une carte à puce et son terminal hôte. Ce protocole met en œuvre une suite d'échanges généralement propriétaires (commandes d'un type dit "APDU" qui seront explicitées ci-après) afin de réaliser le chargement d'une "applet".

La figure 1A annexée à la présente description illustre de façon schématique l'architecture mise en œuvre pour le chargement d' "applets" dans une carte à puce selon l'art connu.

Le terminal 1 stocke un premier programme spécifique de chargement ("Off-Loader"), référencé OL. Il communique avec une carte à puce 2, via un lecteur de carte à puce 3. Les transmissions s'effectuent selon un protocole de communication normalisé, faisant appel aux commandes précitées, protocole qui sera détaillé ci-après.

La carte à puce 2, pour sa part stocke un second programme spécifique de chargement ("In-Loader"), référencé IL.

Un premier inconvénient de ce procédé est que les programmes IL et OL doivent être appariés pour pouvoir communiquer entre eux. Il s'ensuit

que s'ils sont d'origines différentes, ils ne sont pas, *a priori*, compatibles. Cette caractéristique est liée au jeu de commandes à utiliser.

Un deuxième inconvénient est dû au fait que les communications s'effectuent selon le protocole ISO 7816 précité. En effet, celui-ci impose une proximité physique entre les programmes OL et IL. Il s'ensuit que le programme OL doit généralement s'exécuter directement sur le terminal 1 et non, par exemple, sur un autre terminal ou un serveur éloigné.

Or, avec l'essor spectaculaire du réseau Internet, un nombre toujours croissant de terminaux sont connectés à ce réseau, notamment pour pouvoir être en liaison avec des serveurs éloignés, de type "WEB". Il serait donc intéressant, par exemple, de pouvoir stocker la partie "Off-Loader" OL des programmes de chargement d' "applets" sur un serveur "WEB" connecté à ce réseau. Les "applets" à charger sur une ou plusieurs carte(s) à puce pourraient d'ailleurs être aussi stockées sur ce serveur ou sur un ou plusieurs autres serveurs de ce type.

Dans l'état actuel de la technique, ce mode opératoire bute sur une double impossibilité. La première a déjà été évoquée : la norme retenue pour les communications entre le terminal et la carte à puce impose *a priori* une proximité physique entre la localisation des programmes "Off-Loader", OL, et "In-Loader", IL.

D'autre part, les transmissions entre deux systèmes, par exemple un terminal et un serveur éloigné, via le réseau Internet, font appel à des protocoles de type Internet. Dans l'état actuel de la technique, il n'est pas possible de réaliser des communications directes entre une carte à puce et le réseau Internet, comme il va l'être explicité également.

Dans le cadre de l'invention, le terme "réseau Internet" englobe, outre le réseau Internet proprement dit, les réseaux privés d'entreprises ou similaires, du type dit "intranet", et les réseaux les prolongeant vers l'extérieur, du type dit "extranet", et de façon générale tout réseau dans lequel les échanges de données s'effectuent selon un protocole du type

Internet. Dans ce qui suit un tel réseau sera appelé de façon générique "réseau Internet".

On va tout d'abord rappeler brièvement l'architecture générale d'un système d'application à base de carte à puce, relié à un réseau Internet, par référence aux figures 1B et 1C.

Un système d'application à base de carte à puce comporte généralement les éléments principaux suivants :

- une carte à puce ;
- un système hôte constituant le terminal précité ;
- un réseau de communication, à savoir le réseau Internet dans l'application préférée ;
- et un serveur d'application connecté au réseau Internet.

La figure 1B illustre schématiquement un exemple d'architecture de ce type. Le terminal 1, par exemple un ordinateur individuel, comporte un lecteur 3 de carte à puce 2. Ce lecteur 3 peut être ou non physiquement intégré dans le terminal 1. La carte à puce 2 comporte un circuit intégré 20 dont des connexions d'entrées-sorties affleurent en surface de son support pour autoriser une alimentation en énergie électrique et des communications avec le terminal 1. Ce dernier comprend des circuits d'accès 11 au réseau Internet *RI*. Ces circuits peuvent être constitués par un modem pour se connecter à une ligne téléphonique commutée ou à une voie de communication à plus haut débit : réseau numérique à intégration de services ("RNIS"), câble ou liaisons par satellite, etc. Les circuits 11 permettent de se connecter au réseau Internet *RI*, directement ou via un prestataire de services Internet ("Internet Service Provider" ou "ISP", selon la terminologie anglo-saxonne). On peut également avoir recours à un système intermédiaire tel qu'un "proxy" ou un système d'isolation dit "firewall" ("pare-feu" ou encore appelé "garde barrière").

Le terminal 1 comprend naturellement tous les circuits et organes nécessaires à son bon fonctionnement, et qui n'ont pas été représentés dans un but de simplification du dessin : unité centrale, mémoires vive et

fixe, mémoire de masse à disque magnétique, lecteur de disquette et/ou de CédéRom, etc.

Habituellement, le terminal 1 est aussi relié à des périphériques classiques, intégrés ou non, tels un écran de visualisation 5, un clavier 6*a* et une souris 6*b*, etc.

Le terminal 1 peut être mis en communication avec des serveurs ou tous systèmes informatiques connectés au réseau *RI*, dont un seul, 4, est illustré sur la figure 1A. Les circuits d'accès 11 mettent le terminal 1 en communication avec les serveurs 4 grâce à un logiciel particulier 10, appelé navigateur "WEB", ou "browser" selon la terminologie anglo-saxonne. Celui-ci permet d'accéder à diverses applications ou fichiers de données répartis sur l'ensemble du réseau *RI*, généralement selon un mode "client-serveur".

Habituellement, les communications sur les réseaux s'effectuent conformément à des protocoles répondant à des standards comprenant plusieurs couches logicielles superposées. Dans le cas d'un réseau *RI* de type Internet, les communications s'effectuent selon des protocoles spécifiques à ce type de communications, qui seront détaillés ci-après, mais qui comprennent également plusieurs couches logicielles. Le protocole de communication est choisi en fonction de l'application plus particulièrement visée : interrogation de pages "WEB", transferts de fichiers, courrier électronique (e-mel, ou "e-mail" selon la terminologie anglo-saxonne), forums ou "news", etc.

L'architecture logique du système comprenant un terminal, un lecteur de carte à puce et une carte à puce, est représentée schématiquement par la figure 1C. Elle est décrite par la norme ISO 7816, qui elle-même comportent plusieurs sous-ensembles :

- ISO 7816-1 et 7816-2, en ce qui concerne les dimensions et le marquage des cartes ;
- ISO 7816-3, en ce qui concerne le transfert de données entre le terminal et la carte à puce ; et

ISO 7816-4, en ce qui concerne la structure du jeu d'ordres et le format des commandes.

Sur la figure 1C, du côté terminal 1, on n'a représenté que les couches répondant à la norme ISO 7816-3, référencées 101, et un gestionnaire d'ordres "APDU" (norme ISO 7816-4), référencé 102. Du côté carte à puce 2, les couches répondant à la norme ISO 7816-3 sont référencées 200 et le gestionnaire d'ordres "ADPU" (norme ISO 7816-4) est référencé 201. Les applications sont référencées A_1, ..., A_i, ..., A_n ; n étant le nombre maximum d'applications présentes sur la carte à puce 2.

Une application, A_i, présente dans la carte à puce 2, dialogue avec le terminal 1 au moyen d'un jeu d'ordres. Ce jeu présente typiquement des ordres d'écriture et des ordres de lecture. Le format des ordres est connu sous l'abréviation anglo-saxonne de "APDU" (pour "Application Protocol Data Unit"). Il est défini par la norme ISO 7816-4 précitée. Une "APDU" de commande est notée "*APDU.command*" et une "APDU" de réponse est notée "*APDU.response*". Les "APDU" sont échangées entre le lecteur de carte et la carte à puce au moyen d'un protocole spécifié par la norme ISO 7816-3 précitée (par exemple en mode caractère : T=0 ; ou en mode bloc : T=1).

Lorsque la carte à puce 2 inclut plusieurs applications distinctes, comme illustré sur la figure 1C, on parle de carte multi-applicative. Cependant, le terminal 1 ne dialogue qu'avec une seule application à la fois. Une application A_i se présente, par exemple, sous la forme d'une "applet" qui peut être enregistrée initialement, ou encore chargée à partir du terminal 1. Pour ce faire, comme illustré par la figure 1A, on a recours à un programme "Off-Loader", OL, enregistré dans le terminal 1 et à un programme "In-Loader", IL, qui forme l'une des applications A_i de la carte à puce 2.

La sélection d'une application particulière A_i est obtenue à l'aide d'une "APDU" du type sélection ("SELECT"). Une fois ce choix effectué, les

"APDU" qui le suivent sont acheminés vers cette application. Une "APDU SELECT" nouvelle a pour effet d'abandonner l'application en cours et d'en choisir une autre. Le sous-ensemble logiciel gestionnaire des "APDU" 201 permet de choisir une application particulière A_i dans la carte à puce 2, de mémoriser l'application ainsi choisie, et de transmettre et/ou recevoir des "APDU" vers et depuis cette application.

En résumé de ce qui vient d'être décrit, la sélection d'une application A_i et le dialogue avec celle-ci s'effectuent par échanges d'ordres "APDU". On suppose que les applications A_i sont des applications conventionnelles, que l'on appellera ci-après "GCA" (pour "Generic Card Application" ou application de carte générique).

Ce mode opératoire explique que les programmes OL et IL doivent être appariés, pour que les ordres "APDU" échangés puissent être compatibles et compris par ces deux applications.

Ces rappels étant effectués, il est à noter que la carte à puce 2 ne peut communiquer directement avec les navigateurs standards du commerce, sauf à modifier le code de ces derniers.

En outre, et surtout, les cartes à puce actuelles, qui par ailleurs sont conformes aux standards et normes rappelés ci-dessus, ont une configuration matérielle et logicielle qui ne permet pas non plus de communiquer directement avec le réseau Internet. En particulier, elles ne peuvent recevoir et transmettre des paquets de données, selon l'un ou l'autre des protocoles utilisés sur ce type de réseau. Il est donc nécessaire de prévoir une pièce de logiciel additionnelle, implantée dans le terminal 1, généralement sous la forme de ce qui est appelé un "plug-in", selon la terminologie anglo-saxonne. Cette pièce de logiciel, qui porte la référence 12 sur la figure 1B, effectue l'interface entre le navigateur 10 et la carte 2, plus précisément les circuits électroniques 20 de cette carte 2.

L'invention vise à pallier les inconvénients des procédés et dispositifs de l'art connu, et dont certains viennent d'être rappelés, tout en répondant aux besoins qui se font sentir.

Selon une première caractéristique de l'invention, les deux programmes de chargement, OL et IL ne sont plus dépendants l'un de l'autre. En d'autres termes, ils n'ont plus à être appariés pour être compatibles.

Selon une deuxième caractéristique de l'invention, la partie OL des programmes de chargement n'a plus à être stockée obligatoirement dans le terminal, c'est-à-dire en relation de proximité physique avec la seconde partie IL. Tout au contraire, le programme OL peut être stocké sur un serveur éloigné, connecté au terminal via un réseau de type Internet.

Pour ce faire, et selon une autre caractéristique de l'invention, la carte à puce se comporte comme un serveur/client de type "WEB" pour le terminal qui lui est associé.

Pour atteindre ce but, on prévoit une couche de logiciel de communication spécifique dans la carte à puce et son pendant dans le terminal. Le terme "spécifique" doit être entendu comme spécifique au procédé de l'invention. En effet, ces couches de communications, dites spécifiques, sont banalisées quelle que soit l'application considérée. Elles n'interviennent que dans le processus d'échanges de données bidirectionnels entre la carte à puce et le terminal, d'une part, et la carte à puce et le réseau, d'autre part.

Les couches logicielles de communication spécifiques comprennent notamment des composants logiciels, dits "agents intelligents", permettant en particulier des conversions de protocoles. Les agents intelligents seront appelés ci-après plus simplement "agents". Il existe des agents appareillés dans les couches de communication spécifiques respectives associées au terminal et à la carte à puce. Selon le procédé de l'invention, il s'établit des sessions entre agents appareillés.

Selon une autre caractéristique, le procédé de l'invention rend possible l'activation d'applications de type conventionnel, c'est-à-dire du type "CGA" précité, localisées dans une carte à puce, sans devoir les modifier en quoi que ce soit.

Pour ce faire, on prévoit un ou plusieurs agents intelligents particuliers dits traducteurs de script, qui reçoivent des requêtes d'un navigateur et les traduisent en ordres "APDU" compréhensibles par l'application de type "CGA". De ce fait, on implante dans la carte à puce une fonction similaire à celle connue par ailleurs sous la dénomination "CGI" dans les serveurs "WEB" classiques. Cette fonction permet de mettre en œuvre une application dans la carte à puce par un protocole Internet de type "HTTP".

Le chargement d'une "applet" dans la carte à puce peut s'effectuer par cette interface "CGI". La partie IL du programme de chargement est considéré comme étant un script de commandes, que l'on appellera "cgi-script", attaché à la fonctionnalité serveur "WEB" offerte par la carte à puce. Les échanges entre les programmes OL et IL peuvent se dérouler avec l'aide de formulaires classiques en langage "HTML" ou "forms" selon la terminologie anglo-saxonne.

Tout en conservant les normes ISO précitées pour les communications entre terminal et carte à puce, via le lecteur de carte à puce, le procédé selon l'invention permet des échanges entre les parties de programmes de chargement OL et IL en faisant appel au protocole de communication Internet "TCP/IP", la partie OL et les "applets" à charger pouvant être stockées en local ou dans un serveur éloigné.

L'invention a donc pour objet principal un procédé de chargement d'une pièce de logiciel dans une carte à puce à partir d'un terminal connecté à ladite carte à puce par l'intermédiaire d'un lecteur de carte à puce permettant des communications selon un premier protocole déterminé, ledit chargement s'effectuant par la mise en œuvre et la coopération de premier

et second programmes de chargement, ledit second programme de chargement étant stocké dans ladite carte à puce, caractérisé en ce qu'il comprend au moins les phases suivantes :

a/ une première phase préliminaire consistant à implanter, dans ladite carte à puce, une première pièce de logiciel, formant une couche protocolaire de communication spécifique ;

b/ une deuxième phase préliminaire consistant à implanter, dans ledit terminal, une seconde pièce de logiciel, formant une couche protocolaire de communication spécifique ;

en ce que lesdites première et seconde pièces de logiciel comprennent en outre au moins une paire de premières entités logicielles appariées, chacune desdites entités coopérant l'une avec l'autre de manière à permettre l'établissement d'une session d'échanges de données bidirectionnels entre au moins ledit terminal et ladite carte à puce, de manière à ce que ladite carte à puce offre la fonctionnalité d'un client/serveur "WEB" ;

en ce qu'il comprend une troisième phase préliminaire consistant à implanter dans ladite carte à puce au moins une deuxième entité logicielle, apte à interpréter une suite d'instructions et à la traduire en une suite d'ordres, de manière à coopérer avec ladite seconde pièce de logiciel spécifique pour que ladite carte à puce offre une fonctionnalité d'interface passerelle dite "CGI", la dite carte à puce comprenant au moins une desdites suites d'instructions associée au dit second programme de chargement ;

et en ce qu'il comprend au moins les étapes suivantes :

1/ ouverture d'une première session d'échanges de données entre au moins ledit terminal et ladite carte à puce, pour la transmission d'une requête pour que ledit premier programme de chargement récupère des données de paramétrage de chargement fournies par ledit second programme de chargement ;

2/ ouverture d'une deuxième session d'échanges de données entre ladite carte à puce et au moins ledit terminal pour transmettre lesdites données de paramétrage de chargement au dit premier programme de chargement, lesdites données de paramétrage comportant une référence aux dites instructions associées au dit second programme de chargement ; et

3/ ouverture d'une troisième session d'échanges de données entre au moins ledit terminal et ladite carte à puce, pour la soumission d'un fichier de chargement prenant en compte lesdites données de paramétrage de chargement, ledit fichier comprenant des données représentant ladite pièce de logiciel à charger; interprétation de ladite suite d'instructions associée au dit second programme de chargement, par mise en œuvre de ladite fonctionnalité "CGI", de manière générer une suite d'ordres transmise au dit second programme de chargement, à exécuter ce programme et à obtenir ledit déchargement de ladite pièce de logiciel.

L'invention va maintenant être décrite de façon plus détaillée en se référant aux dessins annexés, parmi lesquels :

- la figure 1A illustre schématiquement un exemple de réalisation d'une architecture permettant le chargement d'une "applet" dans une carte à puce selon l'art connu ;
- les figures 1B et 1C illustrent les architectures matérielle et logique, respectivement, d'un exemple de système d'application à base de carte à puce connecté à un réseau Internet selon l'art connu, ;
- la figure 2 illustre schématiquement un exemple de système d'application à base de carte à puce selon l'invention, cette dernière agissant en tant que client/serveur "WEB" ;
- la figure 3 est un diagramme d'états d'une session entre des entités logicielles dites agents intelligents, selon un aspect de l'invention ;

- la figure 4 illustre de façon simplifiée l'architecture logique d'un système selon l'invention dans lequel la carte à puce comprend des agents intelligents ;
- la figure 5 illustre de façon simplifiée l'architecture logique d'un système selon l'invention dans lesquels la carte à puce comprend des agents intelligents traducteurs de scripts ;
- la figure 6 illustre schématiquement un exemple de réalisation d'une architecture permettant le chargement d'une "applet" dans une carte à puce selon l'invention ;
- la figure 7 illustre la structure d'un fichier de chargement d'une "applet" pouvant être en œuvre par le procédé selon l'invention ;
- la figure 8 illustre schématiquement les phases principales du procédé de chargement d'une "applet" dans une carte à puce selon un premier exemple de réalisation pratique ;
- la figure 9 illustre schématiquement les phases principales du procédé de chargement d'une "applet" dans une carte à puce selon un deuxième exemple de réalisation pratique ;
- les figures 9 et 10 illustrent deux exemples de formulaires en langage "HTML" utilisables par le procédé de chargement d'une "applet" dans une carte à puce selon l'invention, pour la mise en œuvre des méthodes dites "GET" et "POST", respectivement ; et
- les figures 12A à 12G illustrent plusieurs variantes de réalisation de réalisation d'architectures de systèmes permettant le chargement d'une "applet" dans une carte à puce selon l'invention.

Dans ce qui suit, sans en limiter en quoi que ce soit la portée, on se placera ci-après dans le cadre de l'application préférée de l'invention, sauf mention contraire, c'est-à-dire dans le cas d'un terminal connecté à un ou plusieurs serveurs éloignés via le réseau Internet.

Avant de décrire le procédé d'activation d'applications localisées dans une carte à puce selon l'invention et de détailler une architecture pour sa mise en œuvre, par référence à la figure 2, il apparaît tout d'abord utile de

rappeler brièvement les caractéristiques principales des protocoles de communication sur les réseaux.

L'architecture des réseaux de communication est décrite par diverses couches. A titre d'exemple, le standard "OSI" ("Open System Interconnection"), défini par l' "ISO", comporte sept couches qui vont des couches dites basses (par exemple la couche dite "physique" qui concerne le support de transmission physique) aux couches dites hautes (par exemple la couche dite "d'application"), en passant par des couches intermédiaires, notamment la couche dite de "transport". Une couche donnée offre ses services à la couche qui lui est immédiatement supérieure et requiert de la couche qui lui immédiatement inférieure d'autres services, via des interfaces appropriées. Les couches communiquent à l'aide de primitives. Elles peuvent également communiquer avec des couches de même niveau. Dans certaines architectures, plusieurs couches peuvent être inexistantes.

Dans un environnement de type Internet, les couches sont au nombre de cinq, et de façon plus précise, en allant de la couche supérieure à la couche inférieure : la couche dite d'application ("http", "ftp", "e-mail", etc.), la couche dite de transport ("TCP"), la couche dite d'adressage de réseau ("IP"), la couche dite de liens de données ("PPP", "Slip", etc.) et la couche dite physique.

Si on se reporte de nouveau à la figure 2, à l'exception de couches logicielles de protocole de communication spécifiques, référencées 13 et 23a, respectivement implantées dans le terminal 1 et la carte à puce 2*a*, les autres éléments, matériels ou logiciels, sont communs à l'art connu, et il n'y a pas lieu de les re-décrire de façon détaillée.

Le terminal 1 comprend des circuits 11 d'accès au réseau *RI*, constitués par exemple d'un modem. Ces circuits regroupent les couches logicielles inférieures, C_1 et C_2, qui correspondent aux couches "physique" et de "lien de données".

On a également représenté les couches supérieures, C_3 et C_4, qui correspondent aux couches "d'adressage de réseau" ("IP", dans le cas d'Internet) et de "transport" ("TCP"). La couche supérieure d'application ("http", "ftp", "e-mail", etc.) n'a pas été représentée.

L'interface entre les couches inférieures, C_1 et C_2, et les couches supérieures, C_3 et C_4, est constituée par une couche logicielle généralement appelée "driver couches basses". Les couches supérieures, C_3 et C_4, s'appuient sur cette interface et sont mises en œuvre par l'intermédiaire de bibliothèques de fonctions spécifiques ou bibliothèques réseau 14, avec lesquelles elles correspondent. Dans le cas du réseau Internet, "TCP/IP" est mis en œuvre au moyen de bibliothèques dites de "sockets".

Cette organisation permet à un navigateur 10 de poser des requêtes vers un serveur 4, pour la consultation de pages "WEB" (protocole "HTTP"), pour le transfert de fichiers (protocole "FTP") ou l'envoi de courrier électronique (protocole "e-mail"), ce de façon tout à fait classique en soi.

Le terminal 1 comprend également un lecteur de carte 3, intégré ou non. Pour communiquer avec la carte à puce 2*a*, le lecteur de carte 30 englobe également deux couches basses, CC_1 (couche physique) et CC_2 (couche de lien de données), jouant un rôle similaire aux couches C_1 et C_2. Les interfaces logicielles avec les couches CC_1 et CC_2 sont décrites, par exemple, par la spécification "PC/SC" ("part 6, service provider"). Les couches elles-mêmes, CC_1 et CC_2, sont notamment décrites par les normes ISO 7816-1 à 7816-4, comme il a été rappelé.

Une couche logicielle supplémentaire 16 forme interface entre les couches applicatives (non représentées) et les couches inférieures, CC_1 et CC_2. La fonction principale dévolue à cette couche 16 est une fonction de multiplexage/démultiplexage.

Les communications avec la carte à puce 2*a* s'effectuent selon un paradigme similaire à celui utilisé pour la manipulation de fichiers dans un

système d'exploitation du type "UNIX" (marque déposée) : OUVRIR ("OPEN"), LIRE ("READ"), ECRIRE ("WRITE"), FERMER ("CLOSE"), etc.

Du côté de la carte à puce 2*a*, on retrouve une organisation similaire, à savoir la présence de deux couches basses, référencées CCa_1 (couche physique) et CCa_2 (couche de lien de données), ainsi qu'une couche d'interface 26a, tout à fait similaire à la couche 16.

Selon une première caractéristique de l'invention, on prévoit, de part et d'autre, c'est-à-dire dans le terminal 1 et dans la carte à puce 2*a*, deux couches de protocoles spécifiques : 13 et 23*a*, respectivement.

Dans le terminal 1, la couche spécifique 13 s'interface aux "drivers couches basses" 15, aux bibliothèques 14 des couches réseau, C_3 et C_4, et aux couches protocolaires du lecteur de carte 3, c'est-à-dire les couches inférieures, CC_1 et CC_2, via la couche de multiplexage 16. La couche spécifique 13 permet le transfert de paquets réseaux de et vers la carte à puce 2*a*. En outre, elle adapte les applications existantes telles le navigateur Internet 10, le courrier électronique, etc., pour des utilisations mettant en œuvre la carte à puce 2*a*.

Du côté de la carte à puce 2*a*, on retrouve une organisation tout à fait similaire constituée par une instance supplémentaire de la couche spécifique, référencée 23*a*, pendant de la couche 13.

De façon plus précise, les couches spécifiques, 13 et 23a, sont subdivisées en trois éléments logiciels principaux :

- un module, 130 ou 230*a*, de transfert de blocs d'informations entre les couches 13 et 23*a*, via les couches conventionnelles CC_1, CC_2, CCa_1 et CCa_2 ;
- une ou plusieurs pièces de logiciel, dites "agents intelligents", 132 ou 232*a*, qui réalisent, par exemple, des fonctions de conversion de protocoles ;

- et un module de gestion de la configuration spécifique, 131 et 231*a*, respectivement ; module qui peut être assimilé à un agent intelligent particulier.

Pour simplifier, on appellera ci-après les agents intelligents, "agents", comme il a été précédemment indiqué.

On retrouve donc, dans le terminal 1 et la carte à puce 2*a*, une pile protocolaire de communication entre les deux entités.

Les couches de niveau deux (couches de lien de données), CC_2 et CCa_2, assurent l'échange entre la carte à puce 2*a* et le terminal 1. Ces couches sont responsables de la détection et l'éventuelle correction d'erreurs de transmission. Différents protocoles sont utilisables, et à titre d'exemples non exhaustifs les suivants :

- la recommandation ETSI GSM 11.11 ;
- le protocole défini par la norme ISO 7816-3, en mode caractère T=0 ;
- le protocole défini par la norme ISO 7816-3, en mode bloc T=1 ;
- ou le protocole défini par la norme ISO 3309, en mode trame "HDLC" (pour "High-Level Data Link Control procedure" ou procédure de commande de liaison à haut niveau).

Dans le cadre de l'invention, on utilisera de préférence le protocole ISO 7816-3, en mode bloc.

De façon connue en soi, à chaque couche de protocole, il est associé un certain nombre de primitives qui permettent les échanges de données entre couches de même niveau et d'une couche à l'autre. A titre d'exemple, les primitives associées à la couche de niveau deux sont du type "demande de données" ("*Data.request*") et "envoi de données" par la carte ("*Data.response*"), ainsi que "confirmation de données" ("*Data.confirm*"), etc.

De façon plus spécifique, les couches 13 et 23a sont chargées du dialogue entre la carte à puce 2*a* et l'hôte, c'est-à-dire le terminal 1. Ces

couches permettent l'échange d'informations entre un utilisateur (non représenté) du terminal 1 et la carte à puce 2*a*, par exemple via des menus déroulants sous la forme d'hypertexte au format "HTML". Elles permettent aussi la mise en place d'une configuration adaptée pour l'émission et/ou la réception de paquets de données.

Comme il a été indiqué ci-dessus, les couches comprennent trois entités distinctes.

La première couche, 130 ou 230*a*, est essentiellement constituée par un multiplexeur logiciel. Elle permet l'échange d'informations entre la carte à puce 2*a* et le terminal hôte 1, sous la forme d'unités de données de protocole. Elle joue un rôle similaire à celui d'un commutateur de paquets de données. Ces unités sont émises ou reçues via la couche de niveau deux (couche de liens de données). Ce protocole particulier de communication permet de mettre en communications au moins une paire d' "agents". Le premier agent de chaque paire, 132, est situé dans la couche 13, côté terminal 1, le second, 232*a*, est situé dans la couche 23*a*, côté carte à puce 2*a*. Une liaison entre deux "agents" est associée à une session, que l'on pourra appeler "*S-Agent*". Une session est un échange de données bidirectionnel entre ces deux agents. Si l'une ou l'autre des couches, 13 et 23a, comporte plusieurs agents, les agents d'une même couche peuvent aussi établir de sessions entre eux et/ou avec les modules 131 et 231*a*, qui constituent des agents particuliers.

De façon plus précise, un agent est une entité logicielle autonome qui peut réaliser tout ou partie de fonctions des couches de niveau trois et quatre, en fonction de la configuration mise en œuvre par le terminal 1.

Les agents sont associés à des propriétés ou attributs particuliers. Pour fixer les idées, et à titre d'exemple non limitatif, les six propriétés suivantes sont associées aux agents :

- "hôte" : agent localisé dans le terminal ;
- "carte" : agent localisé dans la carte à puce ;
- "local" : agent ne communiquant pas avec le réseau ;

- "réseau" : agent communiquant avec le réseau (côté terminal) ;
- "client" : agent qui initialise une session ;
- "serveur" : agent qui reçoit une demande de session.

Un agent particulier est identifié par une référence, par exemple un nombre entier de 16 bits (c'est-à-dire compris entre 0 et 65535). Le bit de poids fort (b15 = 1) indique si la référence est locale (communications locales à la carte à puce ou au terminal) ou distante (b15 = 0).

Il existe deux grandes catégories d'agents : les agents de type "serveur", qui sont identifiés par une référence fixe, et les agents de type "client", qui sont identifiés par une référence variable, que l'on peut qualifier d'éphémère, délivrée par le module de gestion de configuration, 131 ou 231a.

Les agents communiquent entre eux à l'aide d'entité dites "unités de donnée de protocole" ou "*pdu*" (pour "protocol data unit", selon la terminologie anglo-saxonne) constituant une référence de destination et une référence de source. On pourrait également appeler cette "*pdu*" particulière "*SmartTP pdu*", en référence au terme anglais "Smart Card" (carte à puce) couramment utilisé. Les "*pdu*" utilisent notamment les références définies ci-dessus.

Une "*SmartTP pdu*", ou plus simplement "*pdu*" ci-après, comporte une référence source, une référence de destination, un ensemble de bits constituant des drapeaux ou "flags" qui précisent la nature de la "*pdu*", et des données optionnelles :

- le drapeau "*OPEN*" (ouvert) est positionné pour indiquer l'ouverture d'une session ;
- le drapeau "*CLOSE*" (fermé) indique la fermeture d'une session ; et
- Le drapeau "*BLOCK*" (verrouillé) indique que l'agent est en attente d'une réponse de son correspondant et suspend toute activité.

On appellera jeton une "*pdu*" qui ne comporte pas de données.

L'entité "*SmartTP*" contrôle l'existence de l'agent destinataire et réalise la commutation d'un paquet vers ce dernier.

Une session agent "*S-Agent*" possède trois états remarquables, à savoir :

- un état déconnecté : aucune session n'est ouverte avec un autre agent
- un état connecté : une session est ouverte avec un autre agent, une session "*S-Agent*" étant identifiée par une paire de références ; et
- un état bloqué, l'agent étant connecté et attendant une réponse de son correspondant.

Le mécanisme d'établissement d'une session "*S-Agent*" est le suivant :

- une nouvelle instance d'un agent client est créée (côté carte à puce ou terminal), cet agent étant identifié par une référence éphémère pseudo-unique ;
- l'agent client émet une "*pdu*" à destination d'un agent serveur (dont la référence est connue par ailleurs) avec le drapeau "*OPEN*" positionné et l'agent client passe à l'état connecté ou bloqué selon la valeur du drapeau "*BLOCK*" ; et
- l'agent serveur reçoit la "*pdu*" avec le drapeau "*OPEN*" et passe à l'état connecté

Une fois une session ouverte, deux agents échangent des données via des "*pdu*".

Le mécanisme de fermeture d'une session est le suivant :

- un agent émet une "*pdu*" avec le drapeau "*CLOSE*" positionné (et qui comporte éventuellement des données ; et
- l'autre agent reçoit une "*pdu*" avec le drapeau "*CLOSE*" positionné (et qui comporte éventuellement des données) et la session "*S-Agent*" passe à l'état déconnecté.

La figure 3 illustre de façon schématique le diagramme d'états des sessions "*S-Agent*", telles qu'elles viennent d'être rappelées.

Les couches 130 et 230*a* gèrent des tables (non représentées) qui contiennent la liste des agents présents, côté terminal hôte 1 et carte à puce 2*a*.

De façon pratique, les agents permettent d'échanger des données (de l'hypertexte, par exemple), mais également de déclencher des transactions réseau, autorisant des communications entre la carte à puce 2*a* et un serveur éloigné 4 (figure 2).

Les modules de gestion de configuration, 131 et 231*a*, respectivement, sont assimilables à des agents particuliers. Par exemple, le module 131, côté terminal hôte 1, gère notamment des informations relatives à la configuration de ce terminal (modes de fonctionnement), liste des autres agents présents, etc. Le module 231*a*, côté carte à puce 2*a*, a des fonctions analogues. Ces deux agents peuvent être mis en communication l'un avec l'autre pour établir une session.

De façon pratique, la carte à puce 2*a* est avantageusement "adressée" par utilisation d'une adresse "URL" (pour "Universal Resource Locator") définissant un rebouclage sur le terminal 1 lui-même, et non un pointage sur un serveur externe. A titre d'exemple, la structure de cette "URL" est habituellement la suivante :

http://127.0.0.1:8080 (1),

dans laquelle 127.0.0.1 est l'adresse "IP" de rebouclage et 8080 est le numéro de port.

La figure 4 illustre de façon simplifiée l'architecture logique d'un système selon l'invention du type représenté sur la figure 2, mais représenté de façon plus détaillée. La carte à puce 2*a* comprend plusieurs agents, dont deux seulement ont été représentés : un agent de type non précisément défini $232a_1$ et un agent $232a_2$, de type dit "WEB". La pile logique comprend, les couches de protocole inférieures, référencées 200*a*, répondant aux normes ISO 7816-3 (figure 2 : CCa_1 et CCa_2), le gestionnaire de commandes "APDU" $201a_1$, et le multiplexeur de paquets 230*a*, ce dernier étant interfacé aux agents, notamment l'agent "WEB" $231a_2$.

Du côté terminal, il existe deux piles, l'une communiquant avec le réseau Internet *RI*, l'autre avec la carte à puce 2*a*. La première pile

comprend les organes 11 (figure 2 : C_1 et C_2) d'accès au réseau (normes OSI 1 et 2) et les couches de protocole "TCP/IP" (figure 2 : C_3 et C_4), référencées 100. Ces dernières couches sont interfacées avec le navigateur "WEB" 10. L'autre pile comprend les couches de protocole inférieures, référencées 101, répondant aux normes ISO 7816-3 (figure 2 : C_1 et C_2), le gestionnaire 102 d'ordres "APDU" et le multiplexeur de paquets 130, ce dernier étant interfacé avec des agents, dont un seul 132, est représenté. Ce dernier, que l'on supposera de "type réseau", peut en outre communiquer, d'une part avec le navigateur 10, via les couches "TCP/IP" 101, d'autre part avec le réseau Internet *RI*, via ces mêmes couches "TCP/IP" 101 et l'organe 11, d'accès au réseau *RI*.

Le gestionnaire d'ordres "APDU" 201*a* est également interfacé avec une ou plusieurs couches de niveau applications, que l'on appellera simplement applications. Ces applications, A_1, ..., A_i, ..., A_n, sont, comme il a été indiqué, des applications de type conventionnel.

En résumé, la fonction client/serveur "WEB", fournie par la carte à puce 2*a*, peut être réalisée par l'association de l'agent "WEB" $232a_1$ dans la carte à puce et de l'agent réseau 132 dans le terminal 1, et par la mise en œuvre de sessions entre agents, comme il a été décrit.

La carte à puce 2*a* présente donc bien la fonctionnalité client/serveur "WEB". En outre, selon une caractéristique du procédé de l'invention, n'importe quelle application conventionnelle, A_1 à A_n, du type "CGA" précité, peut être activée au travers de ce client/serveur "WEB", soit par le navigateur "WEB" 10 présent dans le terminal 1, soit par un navigateur éloigné 4, localisé en un point quelconque du réseau Internet *RI*, par la mise en œuvre de sessions entre agents. Selon le procédé de l'invention, les applications, A_1 à A_n, ne nécessitent pas d'être ré-écrites et sont mises en œuvre telles quelles.

Dans le cadre de l'invention, tout ou partie des applications A_1 à A_n peut être constituée par des "applets", chargées initialement dans une

mémoire non volatile de la carte à puce 2 ou, au contraire, chargées par l'intermédiaire des deux programmes de chargement OL et IL, dont on précisera ci-après la nature et les lieux de stockage possible.

Selon un autre aspect de l'invention, la fonction serveur "WEB" offerte par la carte à puce inclut un mécanisme similaire à la fonction dite "CGI" (pour "Common Gateway Interface" ou "interface de passerelle") implantée dans les serveurs "WEB" classique.

Avant de décrire un exemple d'architecture conforme à l'invention, permettant de réaliser une fonction de ce type, au sein même de la carte à puce, il est utile de rappeler les principales caractéristiques d'un mode de fonctionnement "CGI".

Le "CGI" est une spécification de mise en œuvre, depuis un serveur "WEB", d'applications écrites pour les systèmes d'exploitation "UNIX" (marque déposée), "DOS", ou "WINDOWS" (marque déposée). A titre d'exemple, pour le système d'exploitation "UNIX", la spécification est "CGI 1.1" et pour le système d'exploitation "WINDOWS 95", la spécification est "CGI 1.3".

Toujours à titre d'exemple, une requête "HTTP" pour une adresse "URL", du type :

"http://www.host.com/cgi-bin/xxx.cgi" (2),

dans laquelle "host" se réfère à un système hôte (généralement éloigné), est interprétée par un serveur "WEB" comme l'exécution d'un script de commande, de type "CGI" nommé "xxx" et présent dans le répertoire "cgi-bin" de ce système hôte. Bien que le nom du répertoire puisse être *a priori* quelconque, par convention, c'est le nom donné au répertoire stockant les scripts de type "CGI". Un script est une suite d'instructions du système d'exploitation du système hôte dont le résultat final est transmis au navigateur "WEB" émetteur de la requête précitée. Différents langages peuvent être utilisés pour écrire ce script, par exemple le langage "PERL" (marque déposée).

De façon pratique, la requête est habituellement affichée sur un écran informatique sous la forme d'un formulaire compris dans une page "HTLM". Le langage "HTLM" permet de traduire un formulaire en une adresse "URL". Le formulaire comporte un ou plusieurs champs, obligatoires ou non, qui sont remplis par un utilisateur à l'aide des moyens de saisie habituels : clavier pour le texte, souris pour les cases à cocher ou les boutons dits "radio", etc. Le contenu du formulaire (ainsi qu'éventuellement des informations et instructions dites "cachées") est émis à destination du serveur "WEB". Le code "HTLM" de la page décrit la structure matérielle du formulaire (cadre, graphisme, couleur, et tout autre attribut), ainsi que la structure des champs de données à saisir (nom, longueur, type de données, etc.).

La transmission peut s'effectuer selon deux types de formats principaux. Un premier format utilise la méthode dite "POST" et un second la méthode dite "GET". Une information de type de format est présente dans le code de la page formulaire.

Ce mécanisme n'est cependant pas directement transposable à une carte à puce, même si celle-ci offre la fonctionnalité client/serveur "WEB" conformément à l'une des caractéristiques de l'invention.

On va maintenant décrire un exemple d'architecture permettant d'activer une application quelconque, de type conventionnel, via un serveur "WEB" sur la carte à puce, par référence à la figure 5.

Parmi les agents intelligents, conforme à l'un des aspects de l'invention, on prévoit des agents intelligents particuliers, que l'on appellera ci-après "Agents traducteurs de script" ou de façon abrégée "ATS". Le script est alors interprété par un des agents intelligents. Cette traduction peut être réalisée de différentes manières :

a/ par l'agent "WEB" $232a_1$ lui-même, qui est doté dans ce cas d'une double capacité ;

b/ par un agent script unique capable de traduire l'ensemble des scripts présents dans la carte à puce 2*a* ;

c/ par un agent de script dédié que l'on appellera "ATSD" ci-après (un agent par script) ; ou

d/ par un agent "APDU" 2010*a* du gestionnaire d'ordres "APDU" 201a, qui est doté, dans ce cas, d'une double capacité.

L'agent "APDU" 2010*a* est une composante de la couche gestionnaire d'ordres "APDU" 201*a*. Cette dernière est une couche capable de centraliser tous les ordres "APDU" émis et/ou reçus par le système, de sélectionner des applications, parmi A_1 à A_n, mais également d'offrir une interface de type agent intelligent. Elle est donc capable, selon l'une des caractéristiques de l'invention de communiquer avec tous les agents intelligents (via des sessions), que ces agents soient localisés dans l'enceinte 6 ou la carte à puce 2*a*.

Dans le cas c/ ci-dessus, une session est ouverte entre l'agent "WEB" 232a_1 et l'un des agents "ATSD".

La figure 5 illustre un exemple d'architecture pour laquelle les agents traducteurs sont du type "ATSD". Ils sont référencés ATS_1 à ATS_n et associés aux applications A_1 à A_n. L'application sélectionnée étant supposée être l'application A_i, la session s'établit entre l'agent "WEB" 232a_1 et l'agent ATS_i.

Un agent traducteur de script génère une suite d'ordres "APDU". Une session est ouverte entre l'agent traducteur, par exemple l'agent ATS_i, et l'agent "APDU" 2101*a*. Les ordres sont alors émis vers l'agent "APDU" 2101*a*. Le gestionnaire d'ordres "APDU" 210*a* sélectionne l'application "CGA" A_i et lui transmet les ordres "APDU", ordres traduits et donc conventionnels, qu'elle est en mesure de comprendre. Cette application est donc correctement activée, sans avoir à la modifier ou à la réécrire.

Les réponses de l'application A_i sont transmises au gestionnaire d'ordres "APDU" 210*a*, à l'agent "APDU" 2010*a*, puis de nouveau à l'agent ATS_i (et de façon plus générale à l'agent traducteur de script).

Les différents cheminements sont représentés symboliquement sur la figure 5 par des traits pleins reliant les blocs fonctionnels, ou en pointillés à l'intérieur de ces blocs.

Le procédé selon l'invention utilise les deux caractéristiques qui viennent d'être rappelées : fonctionnement de la carte à puce en tant que serveur/client "WEB", incluant une fonction "cgi". Le chargement d'une "applet" dans la carte à puce s'effectue en effet via l'interface "CGI" offerte par celle-ci.

De façon plus précise, selon une caractéristique de l'invention, la partie de programme de chargement IL, localisée dans la carte à puce 2a, est constituée par un script. Il s'agit, par exemple, d'un script associé à l'application référencée A_i sur la figure 5. Ce script est, selon une caractéristique du procédé de l'invention, activé par une requête "HTTP", les échanges entre la partie OL et la partie IL s'effectuant selon le protocole de communication "TCP/IP". Les programmes IL et OL deviennent de ce fait *a priori* compatibles. En outre, il n'est plus nécessaire que la proximité physique soit respectée, comme dans l'art connu (voir figure 1). La partie OL peut désormais être localisée dans le terminal ou, de préférence, dans un serveur éloigné (les liaisons entre le serveur et le terminal s'effectuant suivant le protocole "TCP/IP"), voire, comme il le sera montré, être stockée dans la carte à puce elle-même. La requête "HTTP" précitée est initiée par la partie OL.

Il convient de remarquer que les données adressées à l'agent "WEB" $232a_1$ sont transportées, de façon conventionnelle en soi, sous formes d'ordres "APDU" destinés à l'application particulière constituée par le "Multiplexeur de paquets" 230*a*. Le gestionnaire d'ordres "APDU" 201*a* sélectionne cette application de manière tout à fait similaire aux autres

applications de type "CGA", A_1 à A_n, présentes dans la carte à puce 2*a*. En d'autres termes, le multiplexeur de paquets 230*a* est vu par le gestionnaire d'ordres "APDU" 201*a* comme une application "CGA" ordinaire.

La requête "HTTP" est analysée par l'agent "WEB" $232a_1$ qui détecte une référence à un répertoire particulier, que l'on appellera ci-après par convention "cgi-smart" (par analogie à "cgi-bin"), d'une part, et à une application particulière, IL dans le cas de l'exemple décrit. Le chemin complet est donc, en l'occurrence "cgi-smart/il".

Selon une caractéristique du procédé de l'invention, l'entité "il" ci-dessus désigne un script particulier associé une application également particulière (IL en l'occurrence).

Une session est ouverte entre l'agent traducteur, par exemple l'agent ATS_i, et l'agent "APDU" 2010*a*. L'agent traducteur de script ATS_i génère une suite d'ordres "APDU". Les ordres sont émis vers l'agent "APDU" 2010*a*. Le gestionnaire d'ordres "APDU" 201*a* sélectionne l'application "CGA" A_i (par exemple l'application IL) et lui transmet les ordres "APDU", ordres traduits et donc conventionnels, qu'elle est en mesure de comprendre. Cette application est donc correctement activée.

La réponse de l'application IL (A_i) est transmise en sens inverse au gestionnaire d'ordres "APDU" 201*a*, à l'agent "APDU" 2010*a*, puis de nouveau à l'agent ATS_i (et de façon plus générale à l'agent traducteur de script).

La réponse, constituée par un formulaire en langage "HTLM" reprend le chemin inverse, par la mise en œuvre de sessions entre agents intelligents appariés, pour être re-transmise au terminal 1 et, éventuellement à un serveur éloigné 4 (figure 4), via le réseau Internet *RI*, pour atteindre finalement l'application OL.

La figure 6 illustre schématiquement l'architecture logique permettant le chargement d'une "applet" selon le procédé de l'invention. On retrouve sur ce schéma les blocs matériels constitués par le terminal 1, le

lecteur de carte à puce 3 et la carte à puce 2*a*, communiquant par mise en œuvre du protocole normalisé ISO 7816 précité et l'échange d'ordres "APDU", de manière classique en soi. La partie OL est mise en relation avec la partie IL (sous forme d'un script référencé ILs), par des échanges selon le protocole Internet "TCP/IP", de la manière décrite précédemment, par mise en œuvre des fonctions serveur "HTTP" (référencé SC) et "CGI" de la carte à puce 2*a*.

On doit bien comprendre que, bien que représentés à l'extérieur de la carte à puce 2a pour des raisons de commodité, les blocs SC et ILs sont constitués par différents modules internes de celle-ci, qui ont été décrits par référence à la figure 5.

Par contre, le programme OL n'est pas obligatoirement stocké dans le terminal 1.

On va maintenant détailler un premier exemple de chargement d'une "applet" dans une carte à puce 2*a*, par mise en œuvre de la méthode dite "GET".

On suppose que le fichier de chargement de l' "applet", référencé 7, présente la structure illustrée par la figure 7 : une entête 70, un corps principal 71 constitué par du "Byte Code" en langage "JAVA" et une signature électronique 72. L'entête représente un identifiant d'une application particulière, généralement appelé "Application Identifier" ou simplement "AID". La signature électronique 72 est un mot chiffré avec une clé publique ou privée, obtenu à partir du code 71. L'ensemble du fichier 7 peut également être chiffré, pour des raisons de confidentialité, lorsqu'il s'agit d'applications dites sensibles. Optionnellement, on peut prévoir une ou plusieurs signature(s) électronique(s) supplémentaire(s) non représentée(s).

Les principales étapes du processus sont illustrées schématiquement par la figure 8.

Pendant une première étape, la partie de programme de chargement OL récupère, par une commande de type "GET", un formulaire

de chargement à partir de la carte à puce 2*a*, formulaire en langage "HTML" que l'on appellera arbitrairement "download.html".

Cette récupération est effectuée en consultant une page correspondante dont l'URL est typiquement de la forme suivante :

http://127.0.0.1:8080/download.html (3),

dans laquelle http://127.0.0.1:8080 est l'adresse URL de rebouclage proprement dite, telle qu'elle a été définie par la relation (1), et "download.html" la page "HTML" à obtenir. Cette requête met en œuvre une session entre agents intelligents comme il a été décrit en regard des figures 2 à 4, selon un premier aspect de l'invention. La carte à puce 2*a* joue alors le rôle d'un serveur "WEB".

La carte à puce 2*a* envoie le formulaire "download.html" lors d'une deuxième étape, toujours par ouvertures de sessions entre agents intelligents appariés, selon le procédé de l'invention. Le formulaire obtenu peut être affiché sur un écran 5 par l'intermédiaire du navigateur 10.

Pour fixer les idées, un exemple d'un tel formulaire 8 est illustré par la figure 9. Outre diverses zones graphiques et de textes 80 (titre, etc.), le formulaire comprend des zones d'affichage pour l'entête 70 du fichier de chargement 7, le "Byte Code" 71 et la signature 72. La zone d'affichage 71 est du type dit "TEXTAREA" en langage "HTML" et présente une facilité dite d' "ascenseur" pour l'affichage déroulant de textes longs. Les informations correspondantes, telles qu'elles apparaissent sur la figure 9, sont purement arbitraires. Enfin, on prévoit, de façon classique en soi, un bouton d'envoi "send", référencé 81, et un bouton de remise à zéro "reset", référencé 82. Ces boutons sont à la disposition d'un utilisateur du terminal (non représenté). Le bouton d'envoi 81 permet de valider le formulaire et le retransmet à la carte à puce 2a ("soumettre le fichier de chargement" sur la figure 8) et le bouton de remise à zéro 82 permet d'effacer les informations affichées et de ré-initialiser le formulaire.

Le code "HTML" nécessaire pour programmer un tel formulaire est bien connu en soi et est à la portée de l'homme de métier. Il n'est pas

nécessaire de le détailler de nouveau. On peut cependant indiquer qu'il contient notamment une ligne de code en langage "HTML" qui se présente typiquement sous la forme :

<form action="http://127.0.01:8080/cgi-smart/loader"> (4),

dans laquelle http://127.0.01:8080 est l'URL de rebouclage de la relation (1), cgi-smart le répertoire "CGI" précité contenant le script de chargement "loader" que l'on a appelé "il", script associé à la partie IL du programme de chargement.

Si l'affichage visuel du formulaire 8 sur l'écran 5 n'est pas souhaité (par exemple s'il n'y a pas d'opérateur humain, les informations peuvent être cachées incorporant le paramètre "HTML" suivant : "TYPE=hidden" dans la ligne de code (4) précitée.

Lors d'une troisième étape, la partie OL envoie une requête "HTTP" de type "GET" à la carte à puce 2*a*, toujours par ouverture de sessions entre des agents intelligents appariés. En faisant appel à la fonction "CGI" offerte par la carte à puce 2a, telle qu'elle a été décrite en regard de la figure 5, l'application IL s'exécute, le serveur "WEB" formé par la carte à puce 2*a* passant les paramètres de la requête "HTTP" à cette dernière application.

La requête précitée contient une ligne de code typiquement de la forme suivante :

Smart/loader?AID=xxx&ByteCode=yyy&Signature=zzz (5),

Dans laquelle "xxx" est l'entête 70 ("2001" dans l'exemple de la figure 9), "yyy" est le "Byte Code" 71 ("0123456789ABCDEF" dans l'exemple de la figure 9) et "zzz" la signature électronique 71 ("0123456789ABCDEF" dans l'exemple de la figure 9). Les trois parties du fichier de chargement sont donc bien insérées dans trois champs du formulaire "HTML" 8, sous forme concaténée.

Le chargement d'une "applet" particulière identifiée par l'entête 70 a lieu à ce moment.

Enfin, lors d'une quatrième étape un code de retour est transmis de la partie IL à la partie OL, toujours par mise en œuvre de sessions entre

agents appariés. Il s'agit en général d'un simple acquittement ou, si l'opération ne s'est pas réalisée correctement d'un code d'erreur. Dans ce dernier cas, il est nécessaire de renouveler les étapes 1 à 4.

Comme solution alternative, il est possible d'utiliser la méthode "POST" précitée. Pour fixer les idées, la figure 10 illustre un exemple d'un tel formulaire référencé 8'. On retrouve diverses zones de texte et de graphique 80, une zone d'affichage de l'entête 70 et une zone d'affichage de la signature électronique 72, ainsi que des boutons d'envoi "send" 81 et de remise à zéro "Reset" 82. Ces éléments jouent un rôle tout à fait similaire aux éléments de mêmes référence de la figure 9 et il est inutile de les re-décrire. Par contre, la zone d'affichage 71' ne visualise plus explicitement le "Byte Code", mais un répertoire ou un sous-répertoire où le code de l' "applet" à charger est enregistré. En l'occurrence, cette zone pointe sur un fichier, appelé arbitrairement "APPLET.BIN", enregistré sur une unité de stockage appelée "C", qui peut être un disque dur présent dans le terminal 1. Un bouton supplémentaire de navigation "browse" 83 permet de balayer les divers (sous-)répertoires de ce disque et de sélectionner un fichier particulier ("APPLET.BIN").

La méthode "POST" comme la méthode "GET" est bien connue en soi, et il est inutile de la re-décrire en détail. Dans le cadre précis de l'invention, l' "applet" correspondant au fichier "APPLET.BIN" est chargée, à partir de l'unité "C", de manière similaire à ce qui a été décrit pour la méthode "GET".

On va maintenant décrire un deuxième exemple de chargement d' "applet" dans la carte à puce 2*a*.

Il est également possible d'enchaîner plusieurs formulaires lors du chargement. A la place d'un simple statut (acquittement ou code d'erreur dans le premier exemple décrite en regard de la figure 8), le retour de la partie IL contient alors un nouveau formulaire. Ainsi, des séquences dynamiques d'échanges entre les parties OL et IL peuvent être réalisées.

Par exemple, après analyse du fichier de chargement, la partie IL peut demander une autorisation (c'est-à-dire une signature électronique) supplémentaire, par exemple celle d'une instance gouvernementale. Elle renvoie alors à la OL un formulaire qui peut avoir typiquement la structure "HTML" suivante (6) :

```
<TITLE>Authorisation form </TITLE>
<FORM ACTION="http://@carte:8080/cgi-smart/loader">
<INPUT TYPE="text" NAME="GouvSignature" MAXLENGTH="8">Signature
</FORM>
```

dans laquelle "Authorisation form", entre les balises "HTLM" dénommées "<TITLE>" et </TITLE>, représente le titre (arbitraire) du formulaire, "@carte" la traduction littérale de l'adresse URL de rebouclage de la relation (1) et 8080 le numéro de port, la ligne de code :

<INPUT TYPE="text" NAME="GouvSignature" MAXLENTH="8">Signature (7)

requiert l'entrée d'une variable appelée arbitrairement "Signature", en mode texte, de longueur maximale 8 octets, et </FORM> est la balise "HTML" indiquant la fin du code de formulaire.

Le processus complet comprend alors deux étapes supplémentaires avant l'étape finale d'acquittement ou de code d'erreur, soit six étapes, comme illustré par la figure 11.

De façon plus générale, le nombre d'allers et retours peut dépendre de paramètres figurant dans l'un ou l'autre des formulaires échangés entre la carte à puce et la partie OL des programmes de chargement.

Jusqu'à ce point, la localisation de la partie OL n'a pas été précisée expressément. Outre le fait que le procédé rend, a priori, compatible OL et IL, il permet aussi une très grande souplesse précisément quant à cette localisation, étant entendu que la partie IL est stockée dans la carte à puce 2*a* comme formant l'une des applications présente dans cette carte à puce. Le procédé selon l'invention présente notamment l'avantage supplémentaire

de ne plus exiger une proximité physique entre les deux parties OL et IL, puisqu'elles ne sont plus tributaires du protocole de communication ISO 7816, les échanges entre ces deux portions de logiciel mettant en œuvre le protocole de communication Internet TCP/IP.

Aussi, la partie OL, ainsi que les données proprement dites de l' "applet" à charger sur la carte à puce 2a peuvent être stockées soit en local, soit dans un site éloigné. Dans tous les cas cependant, les échanges entre ces deux parties mettent en œuvre, comme il vient d'être rappelé, un protocole de communication "TCP/IP" et le chargement d'une "applet" se déroule comme il a été rappelé précédemment grâce aux fonctions de serveur/client "WEB" et "CGI" offertes par la carte à puce 2*a*.

On va décrire maintenant, par référence aux figures 12A à 12G, les principales architectures pouvant être mises en œuvre dans le cadre de l'invention.

La figure 12 A illustre une architecture de système selon laquelle la partie OL est stockée en local sur le terminal 1. Celui-ci est connecté à un serveur éloigné 4, via le réseau Internet *RI*. Les données de l' "applet" à charger dans la carte à puce 2*a*, référencées *Da*, sont stockées sur ce serveur 4. Une requête "HTTP" permet de les transférer vers la carte à puce 2*a*, via le terminal 1 (et un lecteur de carte à puce non représenté), par mise en œuvre du protocole de communication Internet "TCP/IP".

Dans l'architecture de système représentée sur la figure 12B, la partie de programme de chargement OL et les données *Da* sont stockées localement dans le terminal 1. La connexion du terminal 1 au réseau Internet *RI* est optionnelle. Pour le moins, elle n'est pas requise pour le chargement d'une "applet" selon les étapes du procédé de l'invention. Cette connexion a été représentée en traits pointillés. Le terminal peut donc être autonome.

Dans l'architecture de système représentée sur la figure 12C, la partie de programme de chargement OL et les données *Da* sont stockées dans un serveur distant 4. Les communications entre le serveur 4 et la carte à puce 2*a*, via le réseau Internet *RI*, le terminal 1 et le lecteur de carte à

puce (non représenté) s'effectue par des requêtes "HTTP", et mise en œuvre du protocole "TCP/IP".

L'architecture de système représentée sur la figure 12D est similaire à celle de la figure 12C. La seule différence est que la partie de programme de chargement OL est stockée dans un premier serveur distant, référencé 4*a*, et que les données *Da*, sont stockées dans un second serveur distant, référencé 4*b*.

Dans l'architecture de la figure 12E, la partie du programme de chargement, référencée ici OL', est constituée par un composant du navigateur 10 lui-même. Il s'agit avantageusement d'une "applet" intégrée dans ce navigateur. Le type d'entrée à utiliser dans ce cas est "file" (fichier).

De façon avantageuse également, les données *Da*, de l' "applet" à charger sur la carte à puce 2*a*, peuvent être stockées sur un support d'enregistrement de données externe 9, par exemple une disquette comme illustré par la figure 12E. Naturellement d'autres supports sont utilisables : CédéROM, bande magnétique, etc.

Si on utilise la méthode "POST" précitée, il suffit de spécifier la lettre de l'unité dé stockage, par exemple "A" pour la disquette 9, un éventuel chemin (répertoire, sous-répertoires) et le nom du fichier à charger. Pour fixer les idées, le chemin complet pourrait être typiquement :

A:\APPLET.BIN (8)

Du fait de la fonction serveur/client "WEB" offerte par la carte à puce 2*a*, selon une des caractéristiques du procédé de l'invention, le navigateur 10 est à même, contrairement à l'art connu, de communiquer directement avec cette dernière, comme il l'a été montré en regard des figures 2 à 4. Les communications s'effectuent par l'ouverture de sessions entre agents appariés.

L'architecture de système illustrée par la figure 12F est une variante de l'architecture de la figure 12E. Selon cette variante, la partie de programme de chargement OL est stockée dans la carte à puce 2*a* elle-même, sous forme d'une "applet" en langage "JAVA". Par une requête

"HTTP", cette "applet" peut être chargée dynamiquement sur le terminal 1, en OL". Ce chargement s'effectue à l'aide de requêtes posées par le navigateur 10, lors d'étapes préliminaires. Une fois la partie OL chargée, les étapes ultérieures sont communes au cas précédent. Les données *Da* peuvent également être stockées sur un support externe, par exemple une disquette 8.

L'architecture de système de la figure 12G est une variante de celle de la figure 12F. La seule différence est que la partie de programme de chargement OL est stockée sur un serveur éloigné 4, sous forme d'une "applet" en langage "JAVA". Comme précédemment, par une requête "HTTP", cette "applet" peut être chargée dynamiquement sur le terminal 1, en OL". Ce chargement s'effectue à l'aide de requêtes posées par le navigateur 10, lors d'étapes préliminaires. Les autres étapes sont communes au cas précédent.

Il est clair que d'autres variantes d'architecture peuvent être mises en œuvre sans sortir du cadre de l'invention. Il est notamment possible de charger les données *Da* dans le terminal 1 à partir de diverses sources : par exemple à partir d'un autre système informatique, connecté au terminal 1 par un réseau local ou par tous autres moyens télématiques.

A la lecture de ce qui précède, on constate aisément que l'invention atteint bien les buts qu'elle s'est fixés.

La mise en œuvre du chargement d'une "applet" dans une carte à puce par l'interface "CGI" d'un serveur "WEB" logé dans cette carte à puce présente notamment les avantages suivants :

L'utilisation de formulaires en langage "HTML" standardise le chargement et rend les parties de programmes de chargement OL et IL *a priori* compatibles. En effet, comme il a été montré, c'est la partie IL située dans la carte à puce qui décrit, dans les champs du ou des formulaire(s) renvoyé(s), le paramétrage de chargement auquel il s'attend.

Par ailleurs, ce mécanisme de communication entre les parties de programme de chargement OL et IL permet de gérer simplement des séquences d'échanges dynamiques lors du chargement.

L'utilisation des protocoles Internet "HTTP" et "TCP/IP" pour les échanges entre les parties de programme de chargement OL et IL permet de les séparer physiquement. Seul un routage de paquets "IP" est nécessaire sur le terminal. Le chargement peut alors se faire dans un lecteur carte à puce banalisé, puisque l'on conserve le protocole de communication ISO 7816. Le terminal peut être un simple micro-ordinateur standard connecté à Internet.

Selon un aspect avantageux également du procédé de l'invention, les applications stockées dans la carte à puce restent standards, et donc n'ont pas à être ré-écrites. La carte à puce et le terminal eux-mêmes ne nécessitent que peu de modifications pour pouvoir accommoder le procédé de l'invention : ces dernières se résument à l'implantation, dans ces deux unités, d'une couche logicielle de protocole de communication qui a été appelée spécifique, couche logicielle incluant des agents intelligents.

Alternativement, la partie de programme de chargement OL peut être chargée dynamiquement sur le terminal, au travers la carte, à partir de celle-ci ou d'un serveur "HTTP" éloigné.

Un simple navigateur Internet peut être utilisé comme programme de chargement OL.

Il doit être clair cependant que l'invention n'est pas limitée aux seuls exemples de réalisations explicitement décrits, notamment en relation avec les figures 2 à 12G.

D'autre part, en lieu et place du langage "HTML", d'autres langages similaires, adaptés aux protocoles de communication de type "Internet" peuvent être utilisés, notamment le langage "XML".

L'invention concerne aussi un procédé de chargement d'une pièce de logiciel dans une carte à puce à partir d'un terminal connecté à ladite

carte à puce par l'intermédiaire d'un lecteur de carte à puce permettant des communications selon un premier protocole déterminé, le terminal et la carte à puce comprenant des moyens de traitement d'information et des moyens de stockage d'information, ledit chargement s'effectuant par la mise en œuvre et la coopération de premier et second programmes de chargement, ledit second programme de chargement étant stocké dans les moyens de stockage d'information de ladite carte à puce, caractérisé en ce qu'il comprend au moins les phases suivantes :

a/ une première phase préliminaire consistant à implanter, dans les moyens de stockage d'information de ladite carte à puce (2*a*), une première pièce de logiciel (23*a*), formant une couche protocolaire de communication spécifique ;

b/ une deuxième phase préliminaire consistant à implanter, dans les moyens de stockage d'information dudit terminal (1), une seconde pièce de logiciel (13), formant une couche protocolaire de communication spécifique ;

en ce que lesdites première et seconde pièces de logiciel (13, 23*a*) comprennent en outre au moins une paire de premières entités logicielles appariées (132, 232*a*), chacune desdites entités (132, 232*a*) coopérant l'une avec l'autre, grâce auxdits moyens de traitement d'information du terminal et de la carte à puce, de manière à permettre l'établissement d'une session d'échanges de données bidirectionnels entre au moins ledit terminal (1) et ladite carte à puce (2*a*), de manière à ce que ladite carte à puce (2*a*) offre la fonctionnalité d'un client/serveur "WEB" ;

en ce qu'il comprend une troisième phase préliminaire consistant à implanter dans les moyens de stockage d'information de ladite carte à puce (2*a*) au moins une deuxième entité logicielle (ATS_1 - ATS_n), apte à interpréter une suite d'instructions et à la traduire en une suite d'ordres, de manière à coopérer avec ladite seconde pièce de logiciel spécifique (23*a*) pour que ladite carte à puce offre une fonctionnalité d'interface passerelle dite "CGI",

la dite carte à puce comprenant au moins une desdites suites d'instructions associée au dit second programme de chargement (IL) ;

et en ce qu'il comprend au moins les étapes suivantes :

1/ ouverture d'une première session d'échanges de données entre au moins ledit terminal (1) et ladite carte à puce (2*a*) grâce auxdits moyens de traitement d'information du terminal et de la carte à puce, pour la transmission d'une requête pour que ledit premier programme de chargement (OL) récupère des données de paramétrage de chargement fournies par ledit second programme de chargement (IL) ;

2/ ouverture d'une deuxième session d'échanges de données entre ladite carte à puce (2*a*) et au moins ledit terminal (1) grâce auxdits moyens de traitement d'information du terminal et de la carte à puce, pour transmettre lesdites données de paramétrage de chargement au dit premier programme de chargement (OL), lesdites données de paramétrage comportant une référence aux dites instructions associées au dit second programme de chargement (IL) ; et

3/ ouverture d'une troisième session d'échanges de données entre au moins ledit terminal (1) et ladite carte à puce (2a) grâce auxdits moyens de traitement d'information du terminal et de la carte à puce, pour la soumission d'un fichier de chargement (7) prenant en compte lesdites données de paramétrage de chargement, ledit fichier comprenant des données (70, 71, 72) associées à ladite pièce de logiciel à charger (Da) ; interprétation de ladite suite d'instructions associée au dit second programme de chargement (IL), par mise en œuvre de ladite fonctionnalité "CGI", de manière à générer une suite d'ordres transmise au dit second programme de chargement (IL), à exécuter ce programme (IL) et à obtenir ledit déchargement de ladite pièce de logiciel (Da).

REVENDICATIONS

1. Procédé de chargement d'une pièce de logiciel dans une carte à puce à partir d'un terminal connecté à ladite carte à puce par l'intermédiaire d'un lecteur de carte à puce permettant des communications selon un premier protocole déterminé, ledit chargement s'effectuant par la mise en œuvre et la coopération de premier et second programmes de chargement, ledit second programme de chargement étant stocké dans ladite carte à puce, caractérisé en ce qu'il comprend au moins les phases suivantes :

a/ une première phase préliminaire consistant à implanter, dans ladite carte à puce (2*a*), une première pièce de logiciel (23*a*), formant une couche protocolaire de communication spécifique ;

b/ une deuxième phase préliminaire consistant à implanter, dans ledit terminal (1), une seconde pièce de logiciel (13), formant une couche protocolaire de communication spécifique ;

en ce que lesdites première et seconde pièces de logiciel (13, 23*a*) comprennent en outre au moins une paire de premières entités logicielles appariées (132, 232*a*), chacune desdites entités (132, 232*a*) coopérant l'une avec l'autre de manière à permettre l'établissement d'une session d'échanges de données bidirectionnels entre au moins ledit terminal (1) et ladite carte à puce (2*a*), de manière à ce que ladite carte à puce (2*a*) offre la fonctionnalité d'un client/serveur "WEB" ;

en ce qu'il comprend une troisième phase préliminaire consistant à implanter dans ladite carte à puce (2*a*) au moins une deuxième entité logicielle (ATS_1 - ATS_n), apte à interpréter une suite d'instructions et à la traduire en une suite d'ordres, de manière à coopérer avec ladite seconde pièce de logiciel spécifique (23*a*) pour que ladite carte à puce offre une fonctionnalité d'interface passerelle dite "CGI", la dite carte à puce

comprenant au moins une desdites suites d'instructions associée au dit second programme de chargement (IL) ;

et en ce qu'il comprend au moins les étapes suivantes :

1/ ouverture d'une première session d'échanges de données entre au moins ledit terminal (1) et ladite carte à puce (2*a*), pour la transmission d'une requête pour que ledit premier programme de chargement (OL) récupère des données de paramétrage de chargement fournies par ledit second programme de chargement (IL) ;

2/ ouverture d'une deuxième session d'échanges de données entre ladite carte à puce (2*a*) et au moins ledit terminal (1) pour transmettre lesdites données de paramétrage de chargement au dit premier programme de chargement (OL), lesdites données de paramétrage comportant une référence aux dites instructions associées au dit second programme de chargement (IL) ; et

3/ ouverture d'une troisième session d'échanges de données entre au moins ledit terminal (1) et ladite carte à puce (2*a*), pour la soumission d'un fichier de chargement (7) prenant en compte lesdites données de paramétrage de chargement, ledit fichier comprenant des données (70, 71, 72) associées à ladite pièce de logiciel à charger (*Da*) ; interprétation de ladite suite d'instructions associée au dit second programme de chargement (IL), par mise en œuvre de ladite fonctionnalité "CGI", de manière à générer une suite d'ordres transmise au dit second programme de chargement (IL), à exécuter ce programme (IL) et à obtenir ledit déchargement de ladite pièce de logiciel (*Da*).

2. Procédé selon la revendication 1, caractérisé en ce que ledit lecteur de carte à puce (3) et ladite carte à puce (2*a*) comprennent des première et deuxième piles protocolaires pour lesdites transmissions de données selon ledit premier protocole déterminé, définies par la norme ISO 7816, chacune comprenant au moins des couches protocolaires de

communication logicielles (101, 200a), dites basses, de manière à permettre lesdits échanges de données entre ladite carte à puce (2a) et ledit terminal (1), ces couches formant interface avec lesdites première (13) et seconde (23*a*) pièces de logiciel spécifique formant lesdites couches protocolaires de communication spécifiques, respectivement, et en ce que ces pièces de logiciel (13, 23*a*) comprennent chacune deux entités supplémentaires constituées d'un module de transfert de données (130, 230*a*), formant interface avec lesdites couches basses (101, 200*a*) des première et deuxième piles protocolaires, et d'un module de gestion (131, 231*a*), et en ce que lesdites premières entités de chaque paire sont constituées de modules logiciels, dits agents intelligents (132, 232*a*1) établissant lesdites sessions.

3. Procédé selon la revendication 2, caractérisé en ce que ladite suite d'instructions à interpréter associée au dit second programme (IL) de déchargement est constituée par un script et en ce que ladite deuxième entité logicielle est constituée par un module logiciel dit agent intelligent traducteur de script (ATS_1 - ATS_n) fournissant des ordres compréhensibles par ledit second programme de chargement (OL).

4. Procédé selon la revendication 3, caractérisé en ce que ladite première étape comprend l'émission d'une requête de type dit "HTTP" selon un protocole de type Internet par adressage d'une page déterminée en langage "HTML" contenant lesdites données de paramétrage, ladite adresse étant une adresse de type "URL" de rebouclage sur ladite carte à puce (2*a*).

5. Procédé selon la revendication 4, caractérisé en ce que ladite deuxième étape comprend l'envoi par ladite carte à puce (2*a*) d'un formulaire (8, 8') en langage "HTML" et en ce que ledit formulaire (8, 8') comprend au moins une adresse de type dit "URL" de rebouclage sur ladite carte à puce (2*a*) et un chemin menant à un répertoire déterminé contenant ledit

script associé au dit second programme de chargement (IL), de manière à ce que ce dit premier programme de chargement (OL) récupère les dites données de paramétrage.

6. Procédé selon la revendication 5, caractérisé en ce que ladite troisième étape comprend l'envoi d'une requête de type dit "HTTP" à ladite adresse "URL", désignant ledit répertoire contenant ledit script associé au dit second programme de chargement (IL), ladite requête comprenant lesdites données représentant ladite pièce de logiciel à charger (*Da*), l'interprétation dudit script et l'exécution dudit second programme de chargement (OL), de manière à obtenir ledit chargement de la dite pièce de logiciel (Da).

7. Procédé selon la revendication 6, caractérisé en ce que ladite pièce de logiciel (*Da*) est une appliquette écrite en langage "JAVA" (marque déposée).

8. Procédé selon la revendication 7, caractérisé en ce que ledit fichier de chargement (7) est incorporé dans ledit formulaire (8, 8') et comprend une entête (70) identifiant ladite appliquette, des données (71) et au moins une signature électronique (72) obtenue à partir du chiffrement desdites données.

9. Procédé selon la revendication 8, caractérisé en ce qu'il comprend au moins une première étape supplémentaire, réalisée après ladite troisième étape, et en ce que cette première étape supplémentaire consiste en l'ouverture première session supplémentaire d'échanges de données entre ladite carte à puce (2*a*) et au moins ledit terminal (1) pour transmettre un code prédéterminé reçu par ledit premier programme de chargement (OL).

10. Procédé selon la revendication 9, caractérisé en ce que ledit code prédéterminé consiste en un acquittement lorsque lesdites trois premières étapes se sont déroulées correctement ou un code d'erreur dans le cas contraire.

11. Procédé selon la revendication 10, caractérisé en ce qu'il comprend au moins deux étapes supplémentaires, réalisées après ladite troisième étape, comprenant l'ouverture de sessions d'échanges de données bidirectionnels entre ladite carte à puce (2*a*) et au moins ledit terminal (1) pour la transmission d'un formulaire supplémentaire requérant la soumission de données supplémentaires.

12. Procédé selon la revendication 11, caractérisé en ce que lesdites données supplémentaires comprennent une signature électronique supplémentaire.

13. Procédé selon la revendication 12, caractérisé en ce que ledit terminal étant connecté à au moins un serveur éloigné (4) via un réseau de type Internet (*RI*) et par la mise en œuvre de protocole de communication de type Internet, un desdits agents intelligents (132) est associé à un attribut dit de "réseau" permettant des communications avec ledit réseau Internet (*RI*) et en ce que ledit premier programme de chargement (OL) est stocké sur l'un desdits serveurs éloignés (4, 4*a*).

14. Procédé selon la revendication 13, caractérisé en ce que, ledit terminal (1) comprenant un navigateur de type "WEB" (10), ledit premier programme de chargement (OL') est constitué par un composant logiciel dudit navigateur "WEB" (10).

15. Procédé selon la revendication 14, caractérisé en ce que ledit composant logiciel (OL") est obtenu par une étape initiale de chargement dynamique d'une appliquette (OL) écrite en langage "JAVA" et stockée dans ladite

carte à puce (2*a*), ledit chargement étant obtenu par l'émission d'une requête de type "HTTP", avec un adressage de type "URL" de ladite carte à puce (2*a*).

16. Procédé selon la revendication 17, caractérisé en ce que ledit composant logiciel (OL") est obtenu par une étape initiale de chargement dynamique d'une appliquette (OL) écrite en langage "JAVA" et stockée dans un desdits serveurs éloignés (4), ledit chargement étant obtenu par l'émission d'une requête de type "HTTP", avec un adressage de type "URL" dudit serveur éloigné (4).

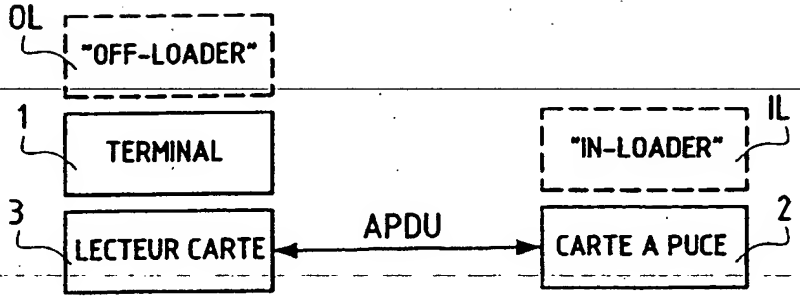


FIG.1A
ART ANTERIEUR

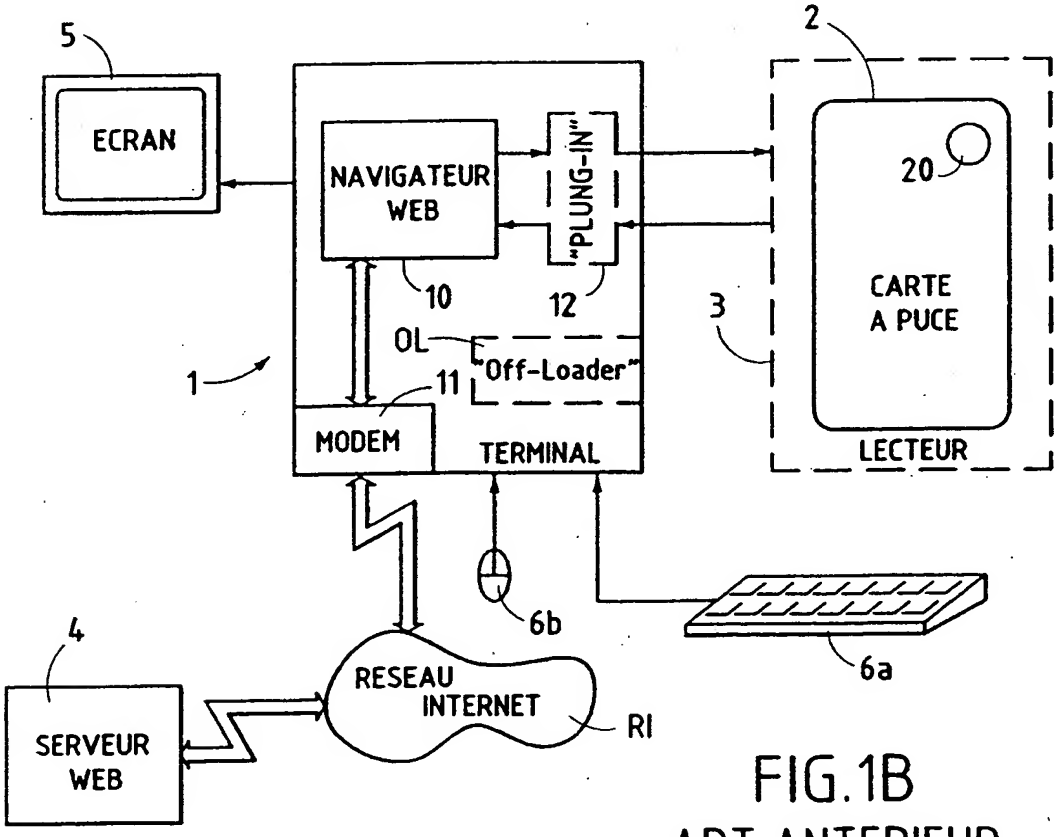


FIG.1B
ART ANTERIEUR

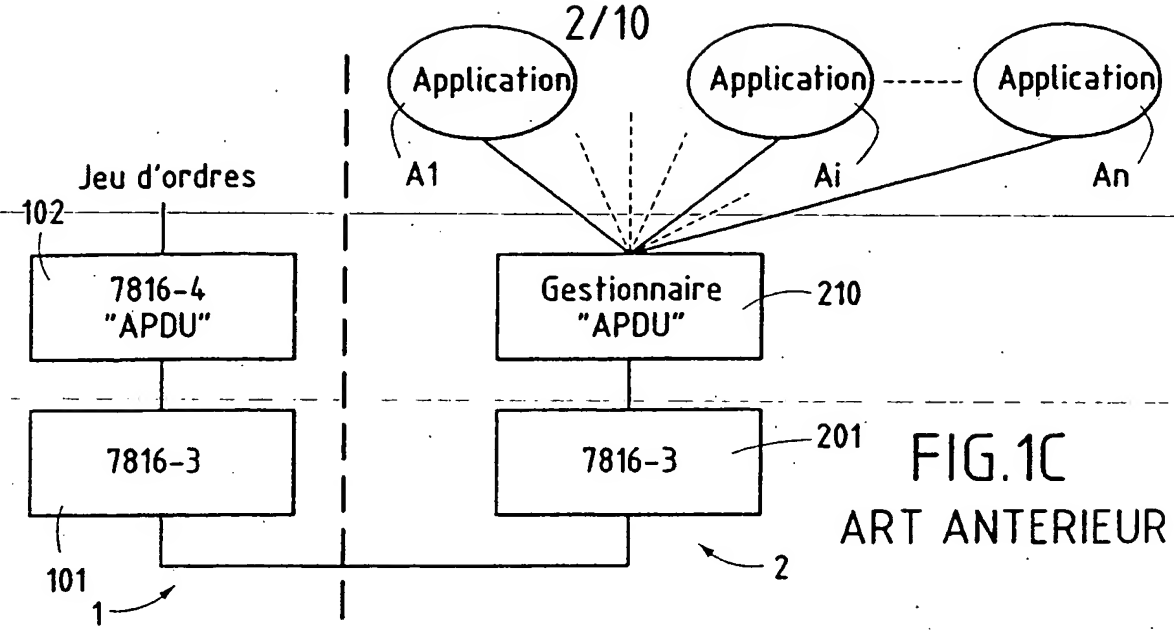


FIG.1C
ART ANTERIEUR

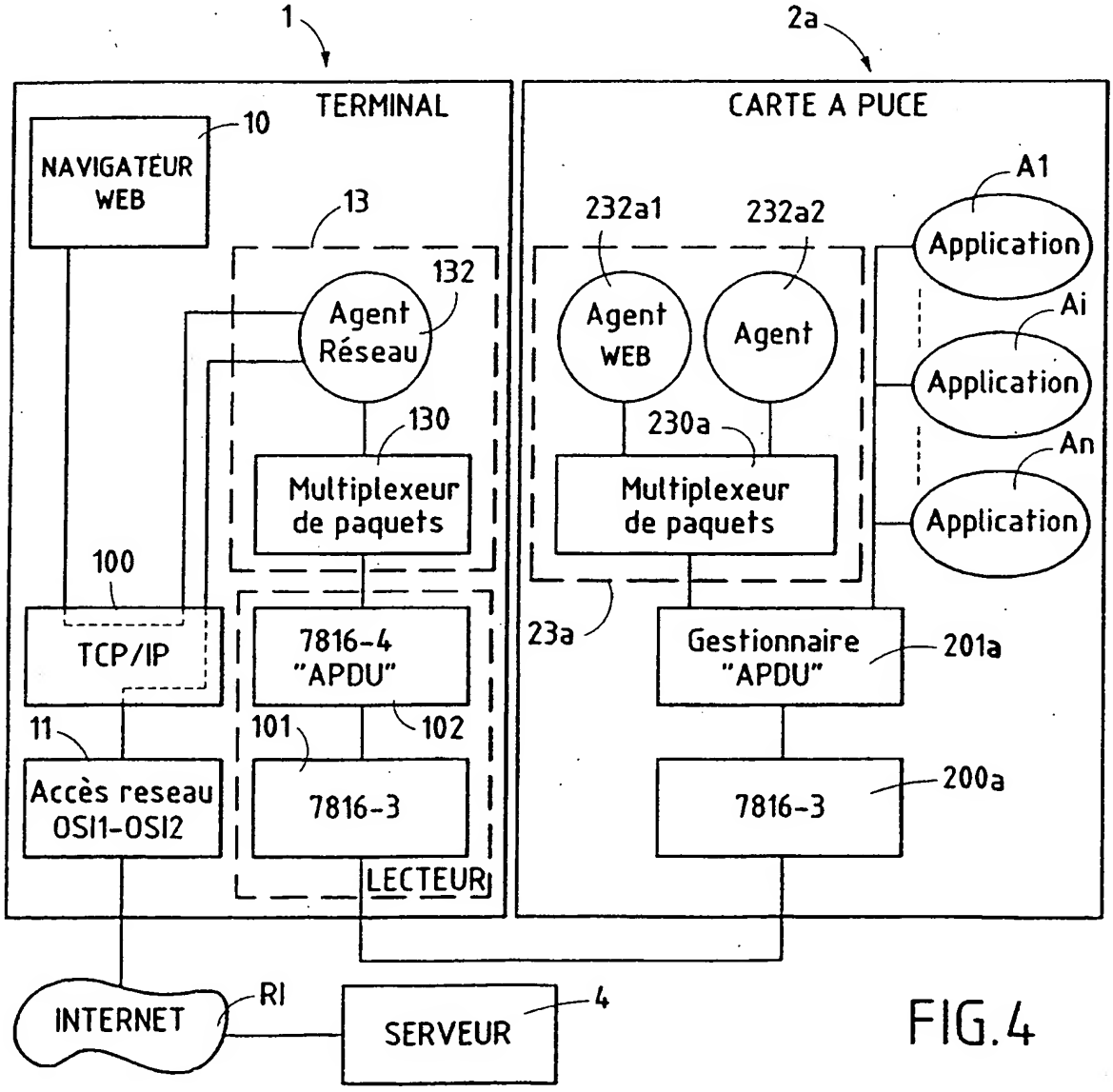


FIG.4

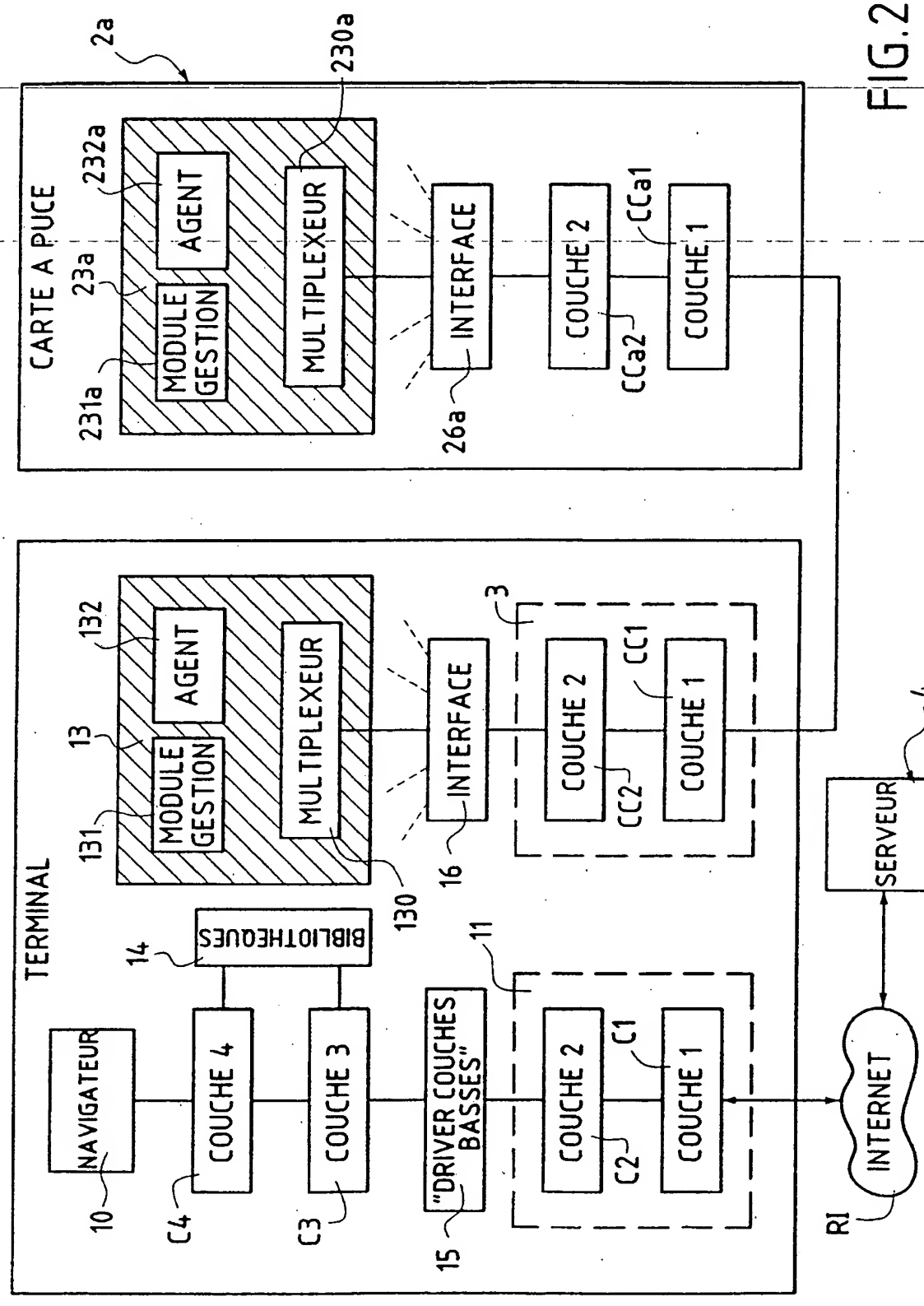
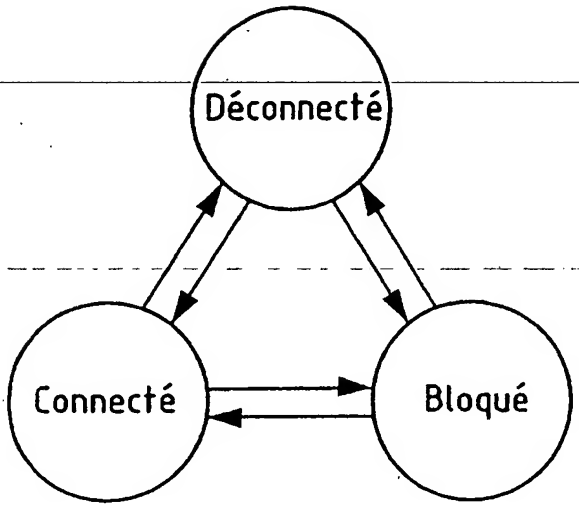

2a
230a
CARTE A PUCE
232a
AGENT
23a
MODULE GESTION
231a
MULTIPLEXEUR
INTERFACE
26a
COUCHE 2
CCa2
CCa1
COUCHE 1
TERMINAL
132
AGENT
13
MODULE GESTION
131
MULTIPLEXEUR
130
INTERFACE
16
3
COUCHE 2
CC2
CC1
COUCHE 1
4
SERVEUR
BIBLIOTHEQUES
14
11
NAVIGATEUR
10
COUCHE 4
C4
COUCHE 3
C3
"DRIVER COUCHES BASSES"
15
COUCHE 2
C2
C1
COUCHE 1
INTERNET
RI
1

FIG.2



Session "S-Agent"

FIG.3

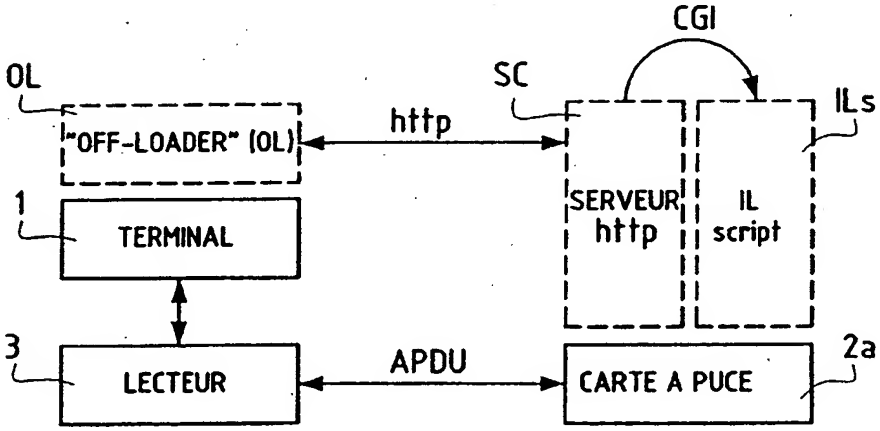


FIG.6

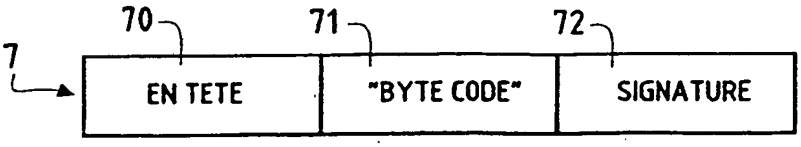


FIG.7

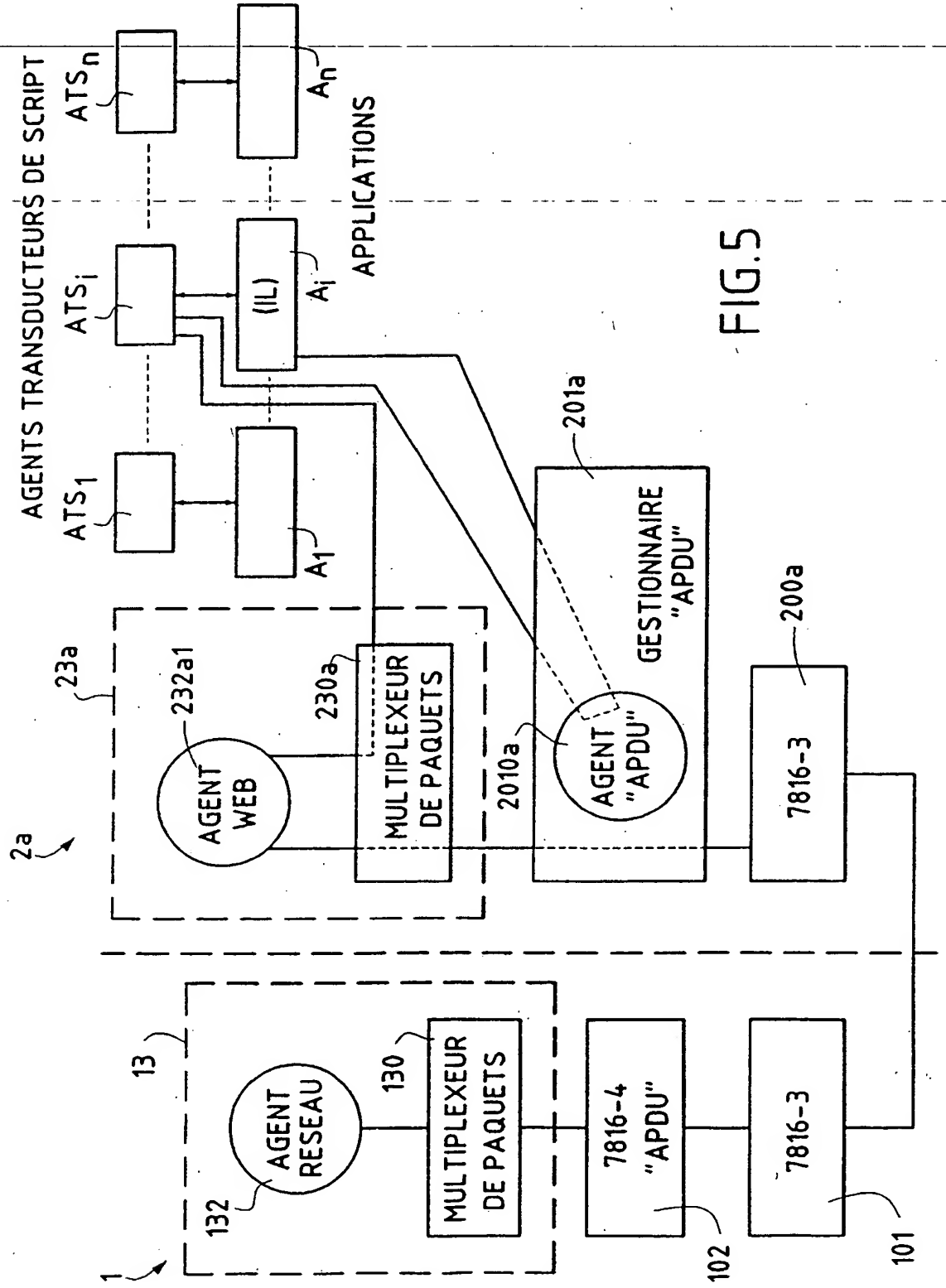

AGENTS TRANSDUCTEURS DE SCRIPT
ATS$_n$
ATS$_i$
ATS$_1$
APPLICATIONS
A$_n$
A$_i$
A$_1$
(IL)
201a
GESTIONNAIRE "APDU"
23a
232a1
AGENT WEB
230a
MULTIPLEXEUR DE PAQUETS
2010a
AGENT "APDU"
200a
7816-3
2a
13
AGENT RESEAU
132
130
MULTIPLEXEUR DE PAQUETS
7816-4 "APDU"
102
7816-3
101
1

FIG.5

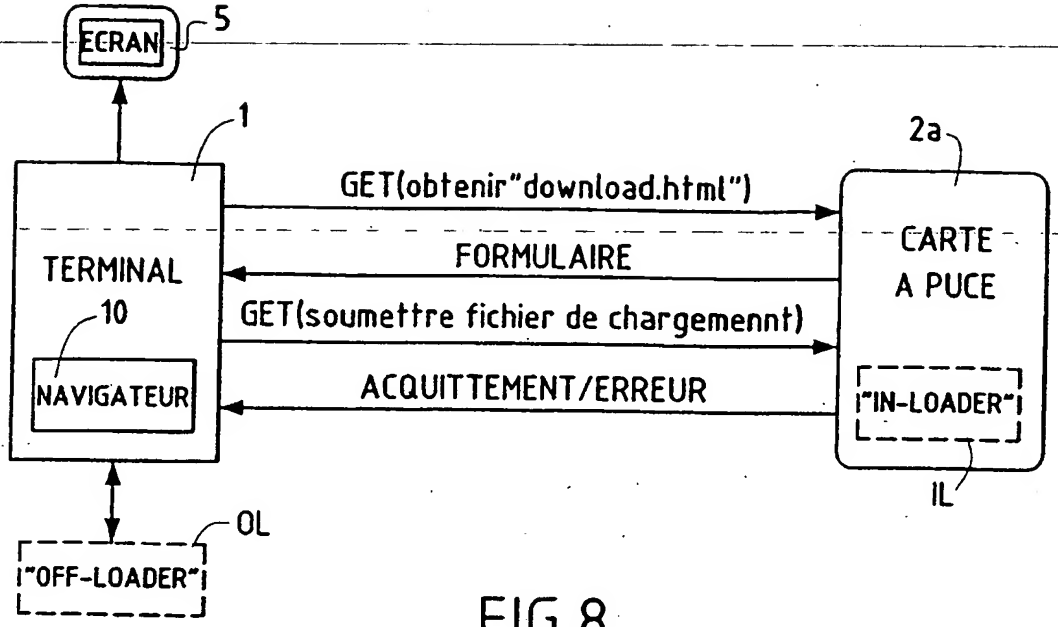

ECRAN
5
1
2a
TERMINAL
10
NAVIGATEUR
GET(obtenir"download.html")
FORMULAIRE
GET(soumettre fichier de chargemennt)
ACQUITTEMENT/ERREUR
CARTE
A PUCE
"IN-LOADER"
IL
OL
"OFF-LOADER"

FIG.8

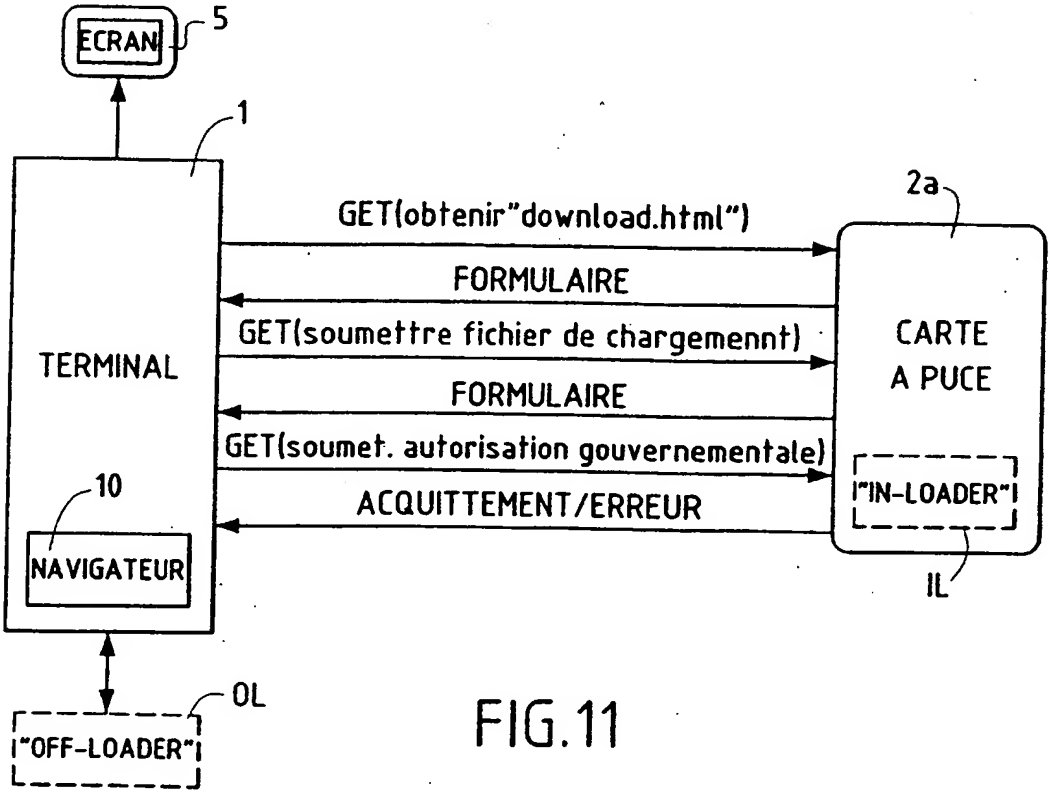

ECRAN
5
1
2a
TERMINAL
10
NAVIGATEUR
GET(obtenir"download.html")
FORMULAIRE
GET(soumettre fichier de chargemennt)
FORMULAIRE
GET(soumet. autorisation gouvernementale)
ACQUITTEMENT/ERREUR
CARTE
A PUCE
"IN-LOADER"
IL
OL
"OFF-LOADER"

FIG.11

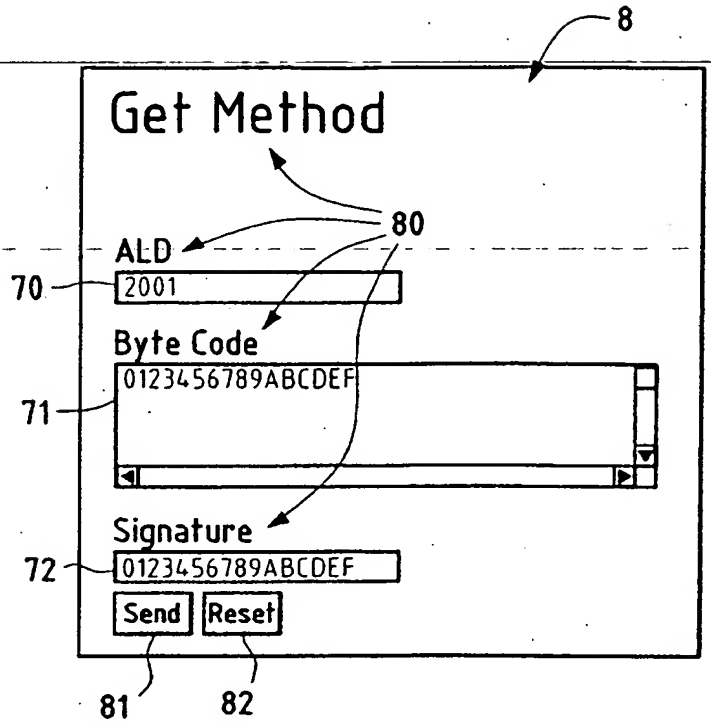

8
Get Method
80
ALD
70
2001
Byte Code
71
0123456789ABCDEF
Signature
72
0123456789ABCDEF
Send
Reset
81
82

FIG.9

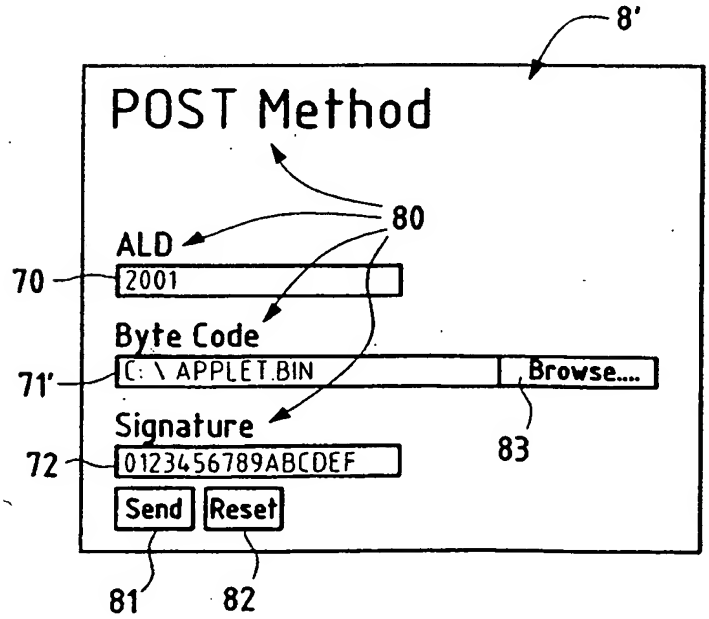

8'
POST Method
80
ALD
70
2001
Byte Code
71'
C: \ APPLET.BIN
Browse....
83
Signature
72
0123456789ABCDEF
Send
Reset
81
82

FIG.10

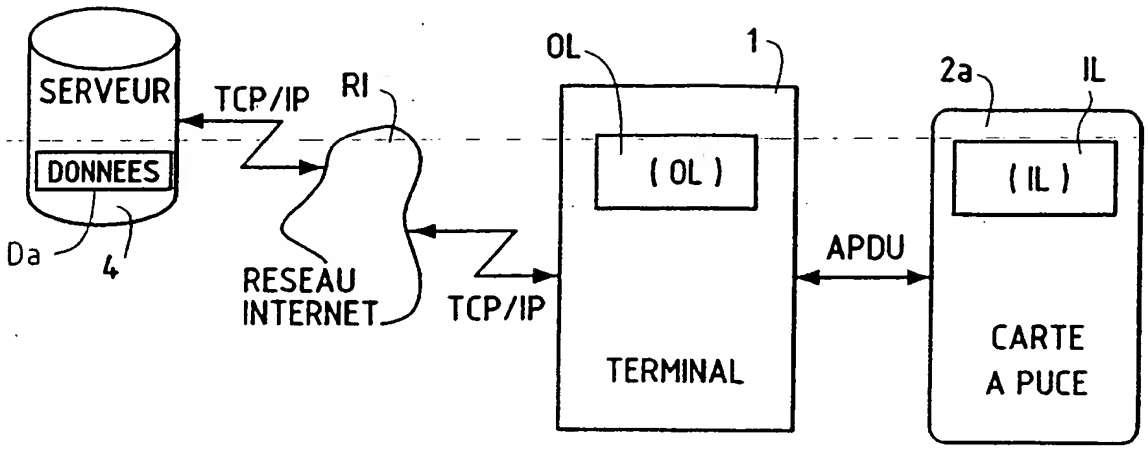


FIG.12A

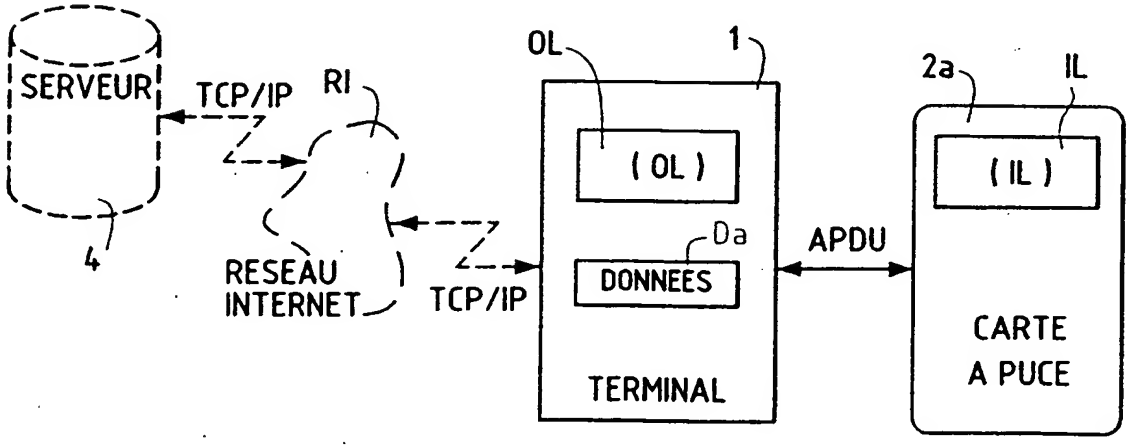


FIG.12B

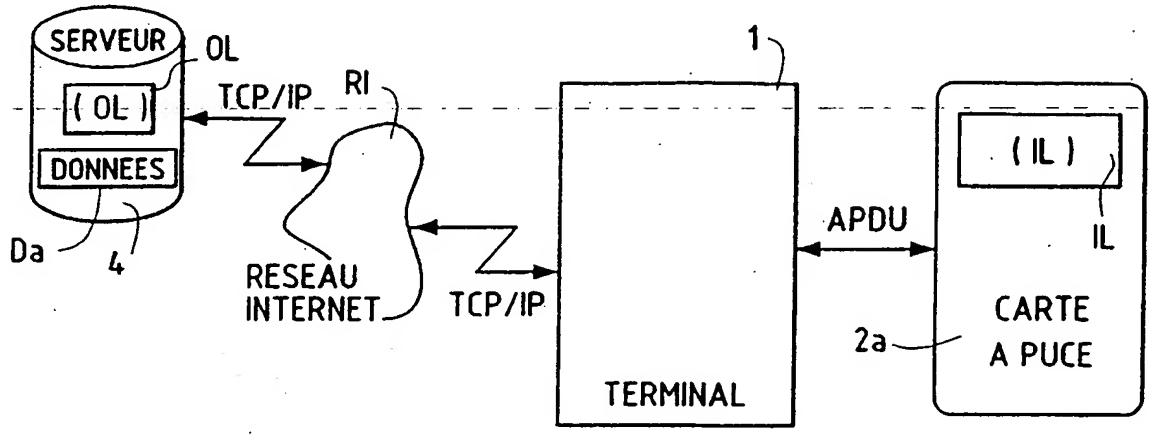


FIG.12C

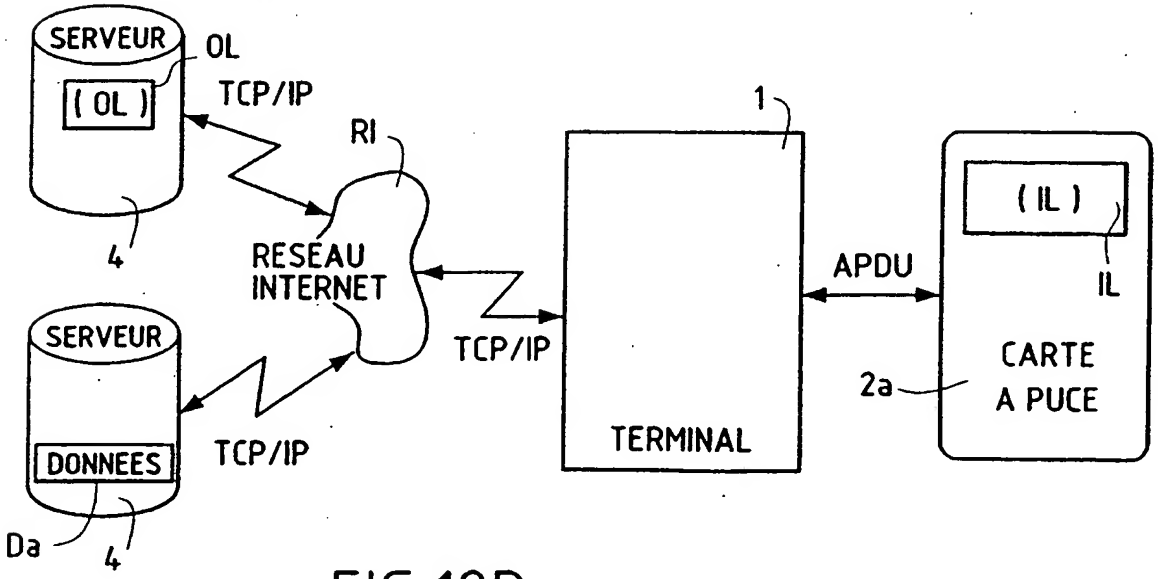


FIG.12D

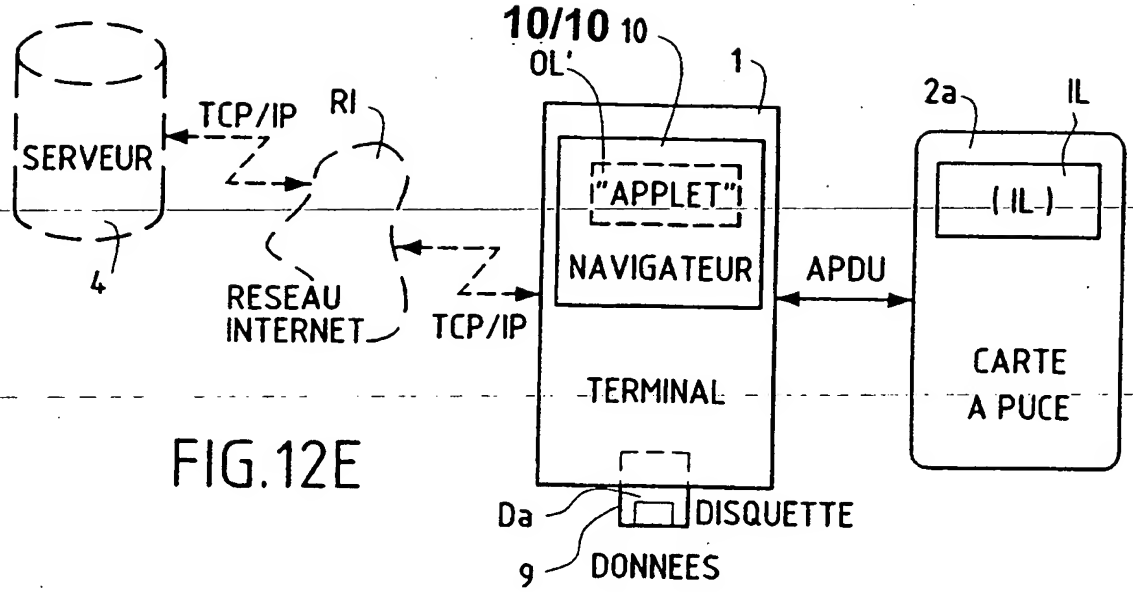


FIG.12E

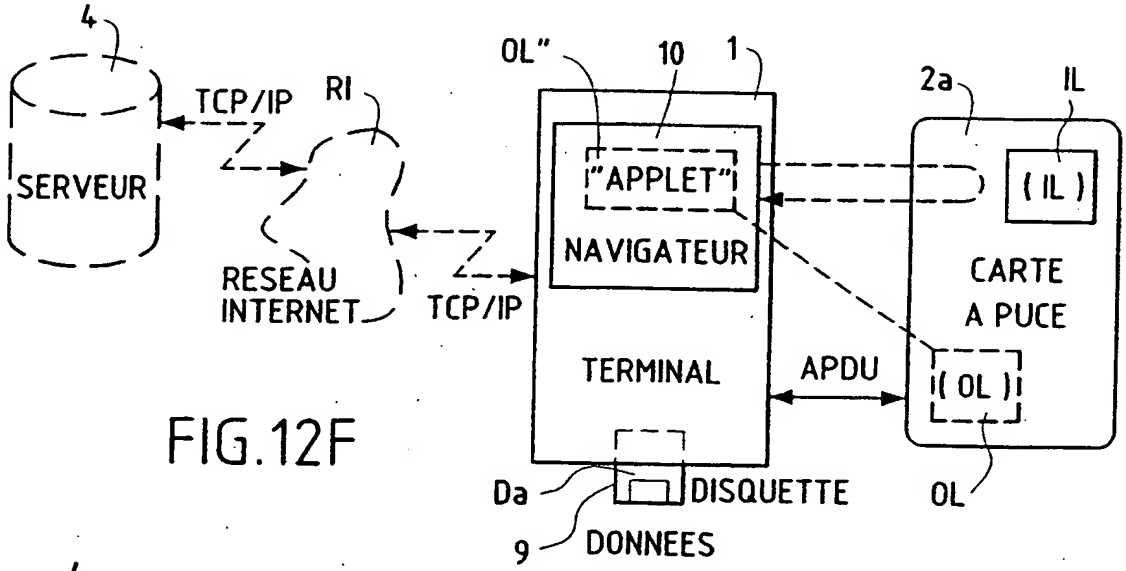


FIG.12F

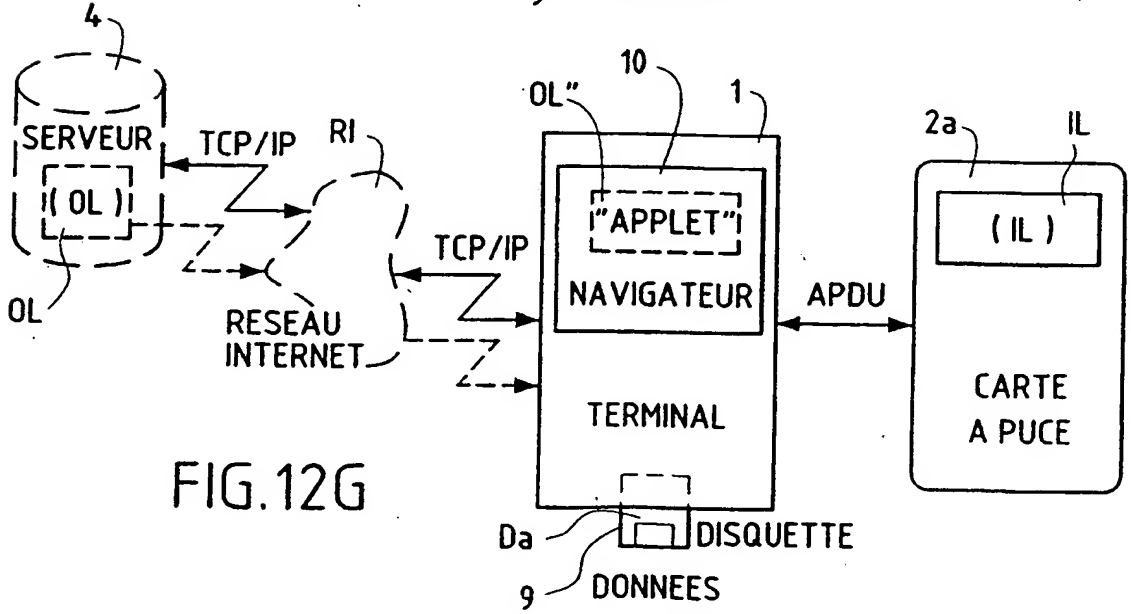


FIG.12G

INTERNATIONAL SEARCH REPORT

Inte .onal Application No
PCT/FR 01/00393

A. CLASSIFICATION OF SUBJECT MATTER
IPC 7 G06F9/445 H04L29/06

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 7 G06F H04L

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)
EPO-Internal

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	WO 98 57474 A (GEMPLUS CARD INT ;MARTINEAU PHILIPPE (FR); MERRIEN LIONEL (US); SI) 17 December 1998 (1998-12-17) abstract; claims 1,3,5,6; figure 2 ---	1,2
Y	WO 98 17029 A (TELIA AB) 23 April 1998 (1998-04-23) the whole document -----	1,2

☐ Further documents are listed in the continuation of box C.

☒ Patent family members are listed in annex.

° Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
13 June 2001	22/06/2001

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Kingma, Y

1

Form PCT/ISA/210 (second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

Information on patent family members

Inte .onal Application No
PCT/FR 01/00393

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
WO 9857474 A	17-12-1998	AU 8113798 A	30-12-1998
		CN 1284230 T	14-02-2001
		EP 1050145 A	08-11-2000
		TW 378308 B	01-01-2000
		ZA 9805151 A	13-04-1999
WO 9817029 A	23-04-1998	SE 506628 C	19-01-1998
		EP 0932956 A	04-08-1999
		NO 991756 A	14-06-1999
		SE 9603825 A	19-01-1998

RAPPORT DE RECHERCHE INTERNATIONALE

Den .e Internationale No
PCT/FR 01/00393

A. CLASSEMENT DE L'OBJET DE LA DEMANDE
CIB 7 G06F9/445 H04L29/06

Selon la classification internationale des brevets (CIB) ou à la fois selon la classification nationale et la CIB

B. DOMAINES SUR LESQUELS LA RECHERCHE A PORTE

Documentation minimale consultée (système de classification suivi des symboles de classement)
CIB 7 G06F H04L

Documentation consultée autre que la documentation minimale dans la mesure où ces documents relèvent des domaines sur lesquels a porté la recherche

Base de données électronique consultée au cours de la recherche internationale (nom de la base de données, et si réalisable, termes de recherche utilisés)
EPO-Internal

C. DOCUMENTS CONSIDERES COMME PERTINENTS

Catégorie *	Identification des documents cités, avec, le cas échéant, l'indication des passages pertinents	no. des revendications visées
Y	WO 98 57474 A (GEMPLUS CARD INT ;MARTINEAU PHILIPPE (FR); MERRIEN LIONEL (US); SI) 17 décembre 1998 (1998-12-17) abrégé; revendications 1,3,5,6; figure 2	1,2
Y	WO 98 17029 A (TELIA AB) 23 avril 1998 (1998-04-23) le document en entier	1,2

☐ Voir la suite du cadre C pour la fin de la liste des documents

☒ Les documents de familles de brevets sont indiqués en annexe

* Catégories spéciales de documents cités:

"A" document définissant l'état général de la technique, non considéré comme particulièrement pertinent

"E" document antérieur, mais publié à la date de dépôt international ou après cette date

"L" document pouvant jeter un doute sur une revendication de priorité ou cité pour déterminer la date de publication d'une autre citation ou pour une raison spéciale (telle qu'indiquée)

"O" document se référant à une divulgation orale, à un usage, à une exposition ou tous autres moyens

"P" document publié avant la date de dépôt international, mais postérieurement à la date de priorité revendiquée

"T" document ultérieur publié après la date de dépôt international ou la date de priorité et n'appartenant pas à l'état de la technique pertinent, mais cité pour comprendre le principe ou la théorie constituant la base de l'invention

"X" document particulièrement pertinent; l'invention revendiquée ne peut être considérée comme nouvelle ou comme impliquant une activité inventive par rapport au document considéré isolément

"Y" document particulièrement pertinent; l'invention revendiquée ne peut être considérée comme impliquant une activité inventive lorsque le document est associé à un ou plusieurs autres documents de même nature, cette combinaison étant évidente pour une personne du métier

"&" document qui fait partie de la même famille de brevets

Date à laquelle la recherche internationale a été effectivement achevée	Date d'expédition du présent rapport de recherche internationale
13 juin 2001	22/06/2001

Nom et adresse postale de l'administration chargée de la recherche internationale	Fonctionnaire autorisé
Office Européen des Brevets, P.B. 5818 Patentlaan 2 NL – 2280 HV Rijswijk Tel. (+31–70) 340–2040, Tx. 31 651 epo nl, Fax: (+31–70) 340–3016	Kingma, Y

1

RAPPORT DE RECHERCHE INTERNATIONALE

Renseignements relatifs aux membres de familles de brevets

Den. .e internationale No
PCT/FR 01/00393

Document brevet cité au rapport de recherche	Date de publication	Membre(s) de la famille de brevet(s)	Date de publication
WO 9857474 A	17-12-1998	AU 8113798 A	30-12-1998
		CN 1284230 T	14-02-2001
		EP 1050145 A	08-11-2000
		TW 378308 B	01-01-2000
		ZA 9805151 A	13-04-1999
WO 9817029 A	23-04-1998	SE 506628 C	19-01-1998
		EP 0932956 A	04-08-1999
		NO 991756 A	14-06-1999
		SE 9603825 A	19-01-1998